

# Application Modernization – End-to-End Example

## From Java EE in 2008 to Quarkus in 2021

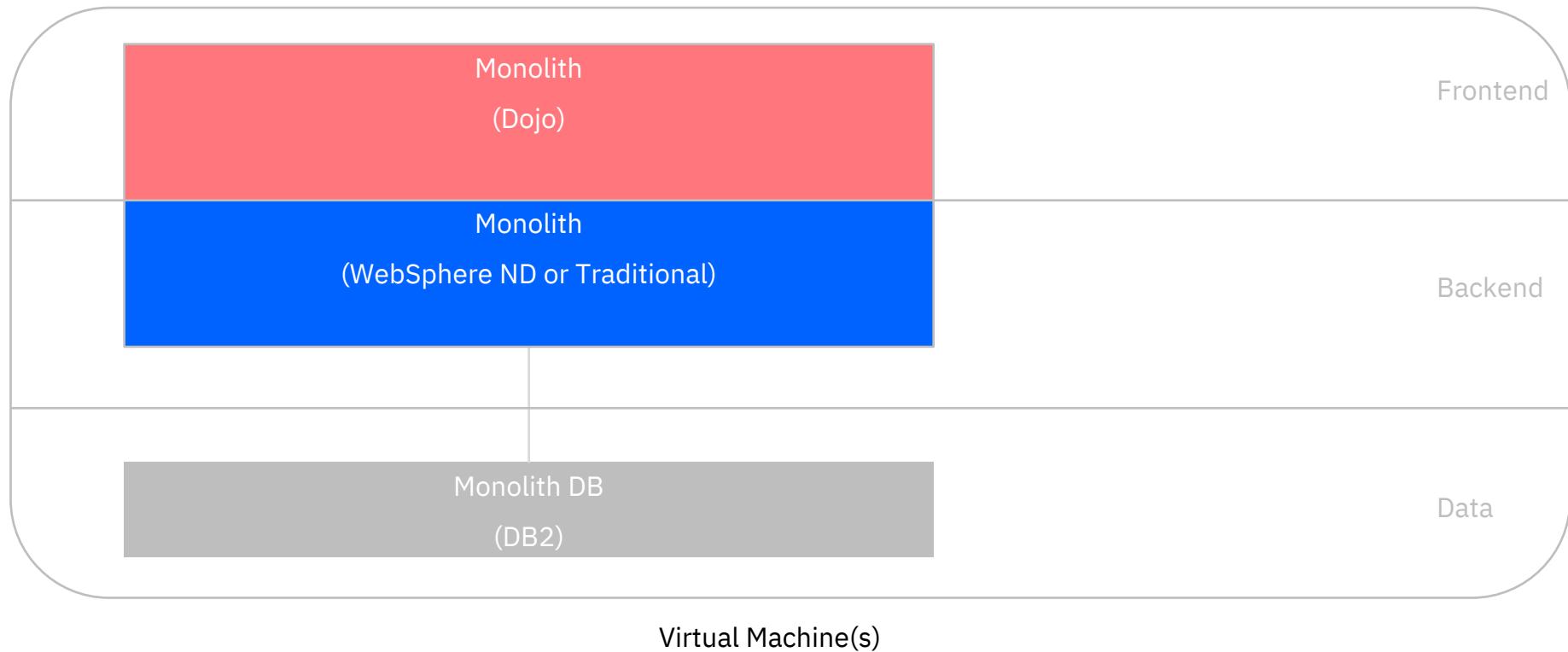
Niklas Heidloff  
Developer Advocate, IBM  
@nheidloff

Draft only  
Work in progress ...

# Step 1 / Starting Point: Monolith running in VMs

Description	How did we get here?	Used Technologies	Containers
Java EE 6 application	N/A – starting point	WebSphere ND or Traditional	None
Built around 2008		EJBs	Two Virtual Machines
WebSphere 8.5.5		REST APIs (BFF, not RESTful)	
Typical three tier architecture		Db2	
No CI/CD		Dojo Toolkit	

# Step 1 / Starting Point: Monolith running in VMs



# Step 1 / Starting Point: Monolith running in VMs

Functionality:  
Browse Products

**Electronic and Movie Depot**

Shop Cart Order History Account

Entertainment ► Electronics ►

**Movies**

Product	Price
Star Wars Episode VI: Return of the Jedi	29.99
Star Wars Episode V: The Empire Strikes Back	29.99
Star Wars Episode IV: A New Hope	29.99
Sony DVD Player	100

Current Shopping Cart

- Empire Strikes Back (2)
- DVD Player (2)
- Return of the Jedi (4)
- BlackBerry Curve (1)
- Sony Ericsson (1)
- New Hope (1)

Order Total: 539.97

# Step 1 / Starting Point: Monolith running in VMs

Functionality:  
Add Products to Shopping Cart

**Electronic and Movie Depot**

Shop    Cart    Order History    Account

Entertainment ▾  
Electronics ▾

**Movies**

Star Wars	Return of the Jedi
Episode VI: Return of the Jedi	50

Star Wars

Return of the Jedi

Current Shopping Cart

- Empire Strikes Back (1)
- Return of the Jedi (1)

Order Total: 59.98

Return of the Jedi

Drag Image To Cart

+

Star Wars

Episode VI:  
Return of the Jedi

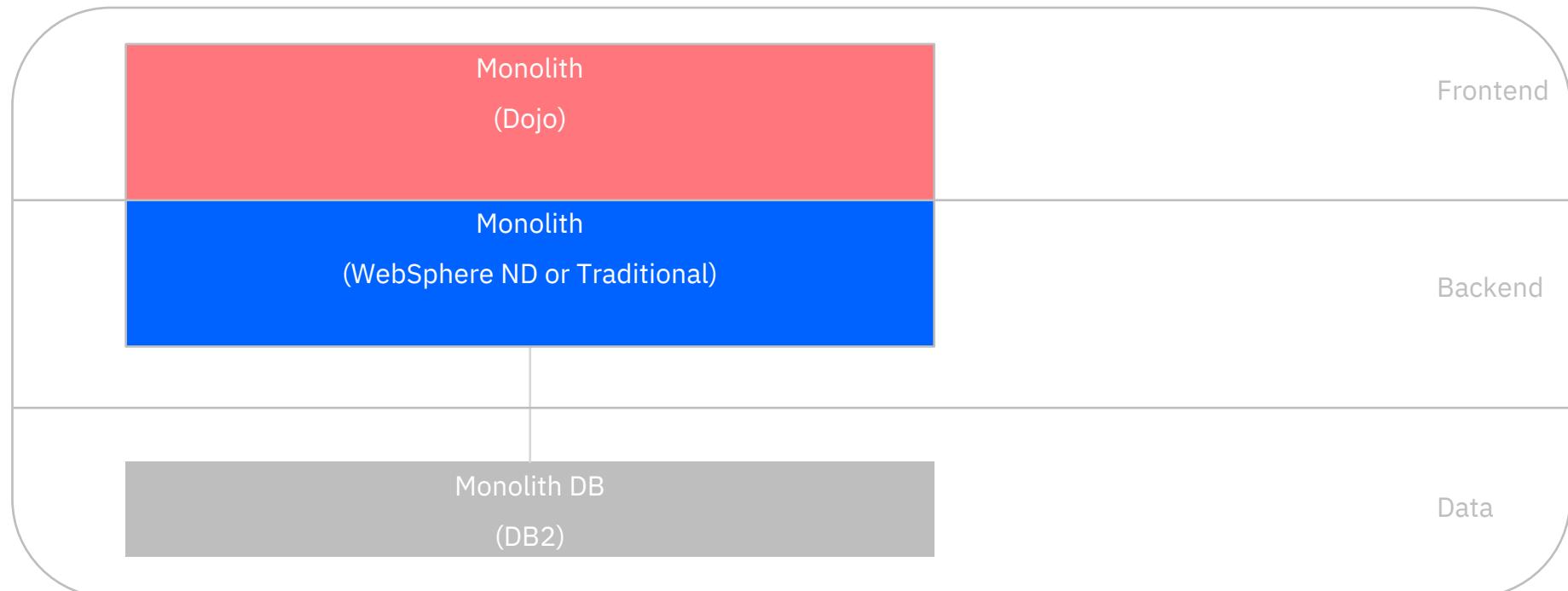
Star Wars

Episode VI:  
Return of the Jedi

# Step 2: Monolith running in Container on WebSphere Traditional

Description and Value	How did we get here?	Used Technologies	Containers
Monolith running in Container  -> Can run on orchestration platforms like OpenShift  -> Consistent management of all applications  -> CI/CD possible	Transformation Advisor generated Dockerfile  Manually created Db2 container	WebSphere Traditional 9	Monolith Db2

# Step 2: Monolith running in Container on WebSphere Traditional



# Step 2: Monolith running in Container on WebSphere Traditional

IBM Cloud Transformation Advisor

Workspace + Home / workspace / collection855

workspace ▾

Collections +/- collection855 +

collection90

collection855

Source environment IBM WebSphere Application Server Profile AppSrv01 Migration target on Cloud Pak for Apps ⓘ Version: 8.5.5.17 WebSphere traditional ⓘ

Upload options ▾

Java applications (2)

Business apps +

0 apps created

Search Filter New Up

Name	Migration Target	Complexity	Issues	Estimated dev cost in days	⋮
CustomerOrderServicesApp.ear	WebSphere traditional	Simple	● ● ● 8	0	⋮
query.ear	WebSphere traditional	Moderate	■ 1 ● 4	1	⋮

# Step 2: Monolith running in Container on WebSphere Traditional

## Migration Files

These files are generated by Transformation Advisor

server\_config.py ↴

install\_app.py ↴

Dockerfile ↴

Pipeline resources ↴

Transformation Advisor generates Dockerfile and scripts to run app in container

```
# Generated by IBM TransformationAdvisor
# Fri Jan 15 13:04:06 UTC 2021

# One manual change in line 12

FROM ibmcom/websphere-traditional:latest-ubi

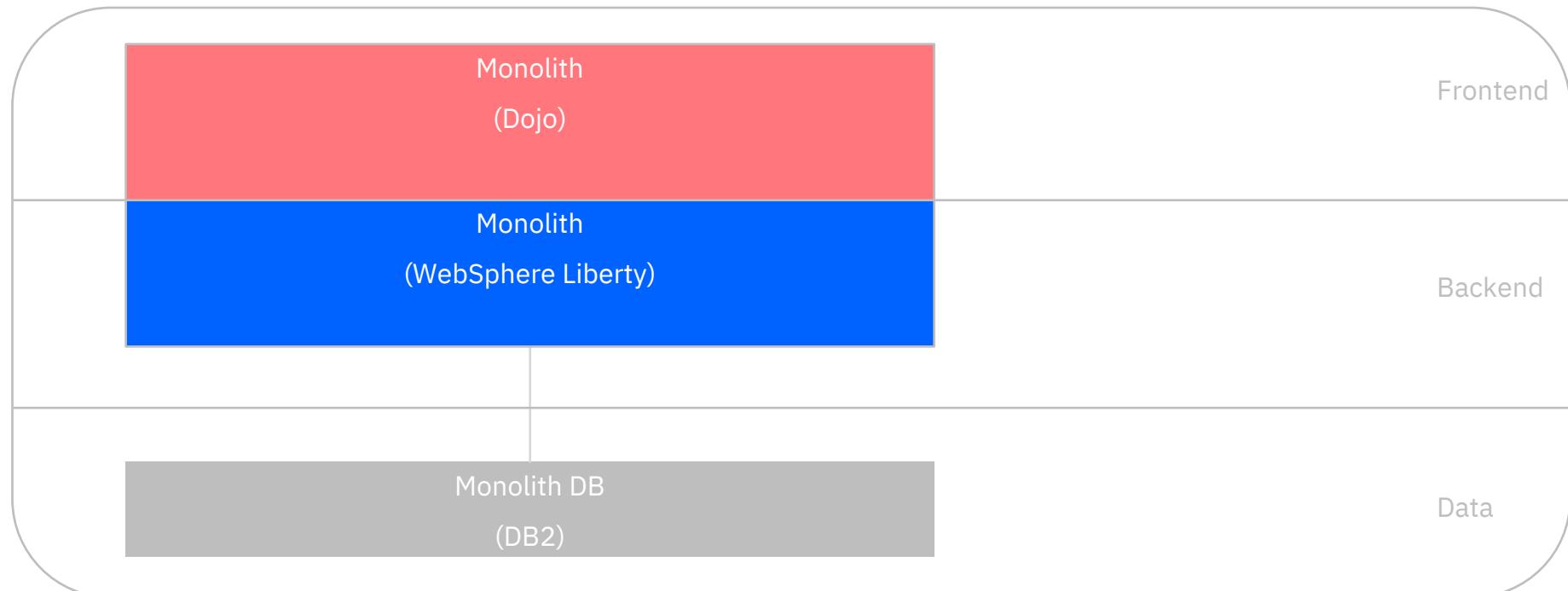
# put app and scripts and properties in /work/config
# put external library (e.g db driver) in /work/config/lib

# Manual fix: Replace {APPLICATION_BINARY} with CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear
COPY --chown=was:root CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear /work/config/CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear
COPY --chown=was:root ./src/config /work/config
COPY --chown=was:root ./lib /work/config/lib
RUN /work/configure.sh
```

# Step 3: Monolith running in Container on WebSphere Liberty

Description and Value	How did we get here?	Used Technologies	Containers
Monolith running on Liberty  -> More efficient runtime for container workloads	Transformation Advisor generated Dockerfile and server.xml  Eclipse Migration Tools for code changes (?)	WebSphere Liberty	Monolith Db2

# Step 3: Monolith running in Container on WebSphere Liberty



# Step 3: Monolith running in Container on WebSphere Liberty

Connectivity Rules Summary											Jump To Rule	▼
This table summarizes the flagged connectivity rules for each Java archive. Select the links in the column header to view all detailed results for that rule. Select the number links within this table to view the detailed results for that specific Java archive.												
	Databases	Enterprise information systems (EIS)	Java EE security	Java Message Service (JMS)	JavaMail server	Message-Driven Beans (MDB)	Remote EJB lookups	Remote EJB providers	Remote web services	Third-party security	Vendor specific messaging	
+ CustomerOrderServicesApp.ear	1		3									

## Detailed Results by Rule

Expand all | Collapse all

### Severe Rules

WebSphere traditional to Liberty

#### 🚫 Behavior change on lookups for Enterprise JavaBeans (4)

Show rule help | Close results

##### Results

FILE NAME	REFERENCE DETAILS	MATCH CRITERIA	LINE NUMBER
CustomerOrderServicesApp.ear/CustomerOrderServicesTest-0.1.0-SNAPSHOT.war			
WEB-INF/ibm-web-bnd.xml		ejb-ref.binding-name=ejblocal:org.pwte.example.service.ProductSearchService	10
CustomerOrderServicesApp.ear/CustomerOrderServicesWeb-0.1.0-SNAPSHOT.war			
WEB-INF/classes/org/pwte/example/resources/CategoryResource.class	Constructor	ejblocal:org.pwte.example.service.ProductSearchService	28
WEB-INF/classes/org/pwte/example/resources/CustomerOrderResource.class	Constructor	ejblocal:org.pwte.example.service.CustomerOrderServices	53
WEB-INF/classes/org/pwte/example/resources/ProductResource.class	Constructor	ejblocal:org.pwte.example.service.ProductSearchService	38

# Step 3: WebSphere Liberty

# Behavior change on lookups for EJBs

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Displays the project structure with files like AbstractCustomer.java and CustomerOrderResource.java.
- Code Editor:** Shows the Java code for CustomerOrderResource.java, with several annotations highlighted in blue.
- Annotations View:** A floating window titled "Java Code Review" displays inspection results:
  - Java EE 7:** 10 results in 14ms
    - CDI [2 results in 10ms]
      - CDI recognizes implicit bean archives [2 results in 1ms]
    - JAX-RS 2.0 [8 results in 0ms]
      - org.codehaus.jackson packages are not available [8 results in 0ms]
        - AbstractCustomer.java:21 org.codehaus.jackson packages are not available
        - Category.java:16 org.codehaus.jackson packages are not available
        - Category.java:17 org.codehaus.jackson packages are not available
        - LineItem.java:19 org.codehaus.jackson packages are not available
        - Order.java:21 org.codehaus.jackson packages are not available
        - Product.java:16 org.codehaus.jackson packages are not available
        - Product.java:17 org.codehaus.jackson packages are not available
        - CustomerServicesApp.java:23 org.codehaus.jackson packages are not available
    - Java EE 8 [1 result in 19ms]
      - JAX-RS 2.1 [1 result in 10ms]
        - The com.ibm.json packages are not available in the jaxrs-2.1 feature [1]
    - Liberty migration [3 results in 32ms]
      - WebSphere traditional to Liberty [3 results in 32ms]
        - Behavior change on lookups for Enterprise JavaBeans [3 results in 0ms]
          - CategoryResource.java:29 Behavior change on lookups for Enterprise JavaBeans
          - CustomerOrderResource.java:53 Behavior change on lookups for Enterprise JavaBeans
          - ProductResource.java:38 Behavior change on lookups for Enterprise JavaBeans
  - Right Panel:** Displays a message about behavior changes on lookups for Enterprise JavaBeans, followed by two examples of flagged code snippets.

# Step 3: WebSphere Liberty

com.ibm.json not available

eclipse-workspace3 - CustomerOrderServicesWeb/src/org/pwte/example/resources/CustomerOrderResource.java - Eclipse IDE

Project Explorer

AbstractCustomer.java CustomerOrderResource.java

```
34 import org.pwte.example.exception.CustomerDoesNotExistException;
35 import org.pwte.example.exception.GeneralPersistenceException;
36 import org.pwte.example.exception.InvalidQuantityException;
37 import org.pwte.example.exception.OrderModifiedException;
38 import org.pwte.example.exception.ProductDoesNotExistException;
39 import org.pwte.example.service.CustomerOrderServices;
40
41 import com.ibm.json.java.JSONArray;
42 import com.ibm.json.java.JSONObject;
43
44 @Path("/Customer")
45 @TransactionAttribute(TransactionAttributeType.NOT_SUPPORTED)
46 public class CustomerOrderResource {
47     CustomerOrderServices customerOrderServices = null;
48
49     public CustomerOrderResource()
50     {
51         try {
52             InitialContext ctx = new InitialContext();
53             customerOrderServices = (CustomerOrderServices) ctx.lookup("ejblocal:org.pwte.ex");
54         } catch (NamingException e) {
55             e.printStackTrace();
56         }
57     }
58 }
```

Markers Properties Servers Data Source Snippets Problems Annotations Software An

XML File Review JSP Code Review File Review Java Code Review

View as: Tree

was-to-lib (21/01/2014:11:26) [24]

Java EE 7 [10 results in 14ms]

- CDI [2 results in 10ms]
  - CDI recognizes implicit bean archives [2 results in 1ms]
- JAX-RS 2.0 [8 results in 0ms]
  - org.codehaus.jackson packages are not available [8 results in 0ms]
    - AbstractCustomer.java:21 org.codehaus.jackson packages are not available
    - Category.java:16 org.codehaus.jackson packages are not available
    - Category.java:17 org.codehaus.jackson packages are not available
    - LineItem.java:19 org.codehaus.jackson packages are not available
    - Order.java:21 org.codehaus.jackson packages are not available
    - Product.java:16 org.codehaus.jackson packages are not available
    - Product.java:17 org.codehaus.jackson packages are not available
    - CustomerServicesApp.java:23 org.codehaus.jackson packages are not available

Java EE 8 [1 result in 19ms]

- JAX-RS 2.1 [1 result in 10ms]
  - The com.ibm.json packages are not available in the jaxrs-2.1 feature [1 result in 10ms]
    - CustomerOrderResource.java:41 The com.ibm.json packages are not available in the jaxrs-2.1 feature

Liberty migration [3 results in 32ms]

- WebSphere traditional to Liberty [3 results in 32ms]
  - Behavior change on lookups for Enterprise JavaBeans [3 results in 0ms]
    - CategoryResource.java:29 Behavior change on lookups for Enterprise JavaBeans
    - CustomerOrderResource.java:53 Behavior change on lookups for Enterprise JavaBeans
    - ProductResource.java:38 Behavior change on lookups for Enterprise JavaBeans

The com.ibm.json packages are not available in the jaxrs-2.1 feature

The json4j com.ibm.json packages were previously included in the jaxrs-1.1 and jaxrs-2.0 Liberty features. However, the packages are no longer included in the jaxrs-2.1 feature.

Note: This rule is flagged once per Eclipse project or Java archive.

**Option 1 (Recommended): Refactor your application to use JSON-P**

It is recommended to switch your application from using JSON4J to using the JSON Processing (JSON-P) APIs provided by the Java EE specification. Once you have made the necessary application changes, no further changes to the Liberty server.xml configuration is required since the jsonp-1.1 feature is enabled by the jaxrs-2.1 feature.

For more information on JSON-P, see [Java API for JSON Processing](#).

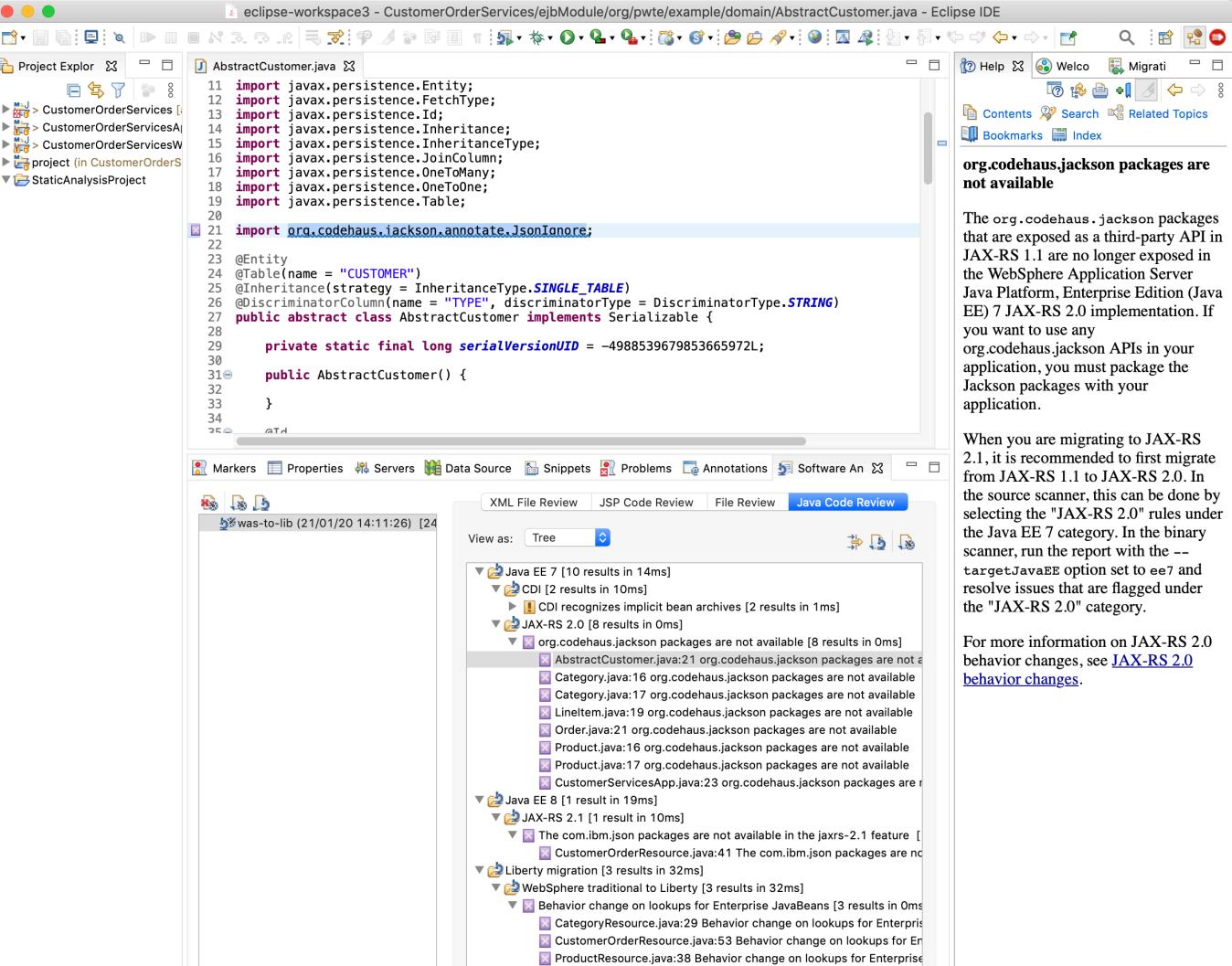
**Option 2: Add the json-1.0 feature in server.xml**

If you want to continue using the json4j com.ibm.json packages, add the json-1.0 feature to your Liberty server.xml configuration file.

For more information on the json-1.0 feature, see [JavaScript Object Notation for Java](#).

# Step 3: WebSphere Liberty

org.codehaus.jackson  
not available



# Step 3: Monolith running in Container on WebSphere Liberty

**Question:**

**Why doesn't TA show all results that Migration Tools show?**

# Step 3: Monolith running in Container on WebSphere Liberty

## Migration Files

These files are generated by Transformation Advisor

server.xml [↓](#)

pom.xml [↓](#)

Operator resources [↓](#)

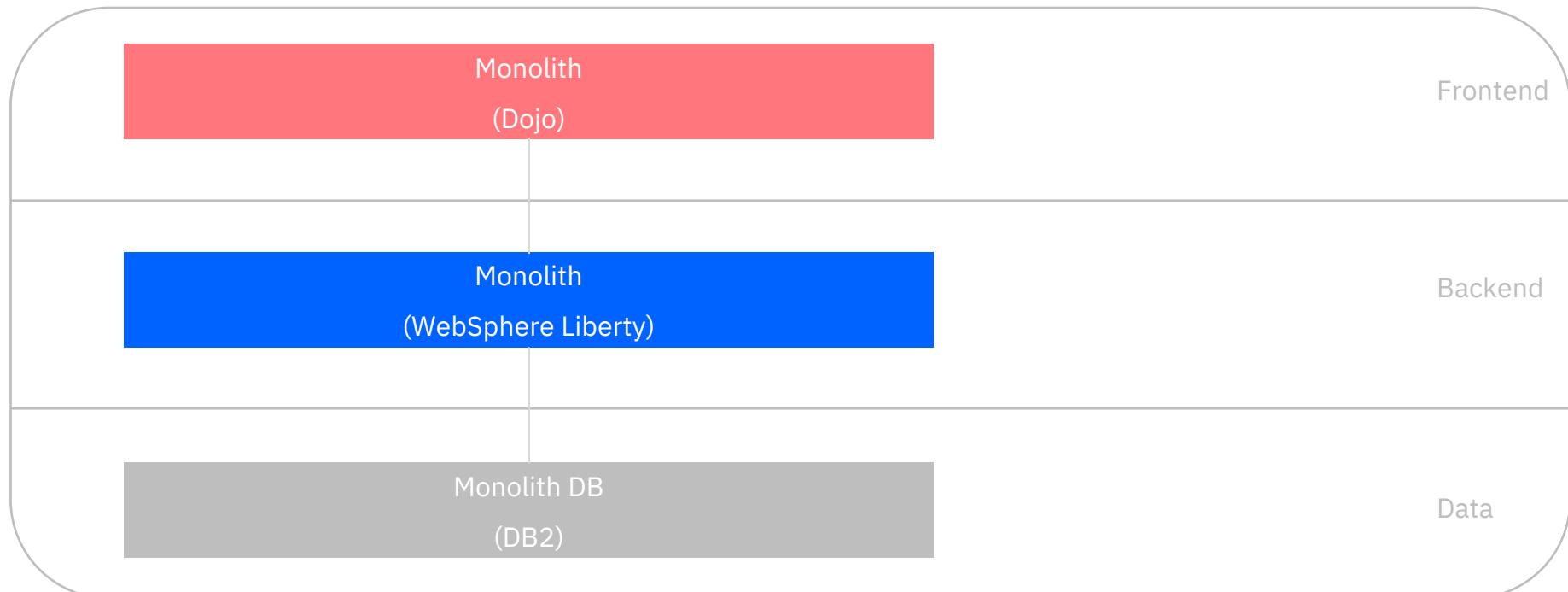
Dockerfile [↓](#)

**Question:**  
**Why don't the generated files work?**

# Step 4: Separated Frontend

Description and Value	How did we get here?	Used Technologies	Containers
Separate container for Dojo frontend  -> Easier extensions and potential replacement of frontend code	Manually	WebSphere Liberty Dojo	Backend monolith Frontend monolith Db2

# Step 4: Separated Frontend



# Step 4: Separated Frontend

For now the sample is run with Docker only

As ‘ingress’ an nginx proxy is used

```
version: '3.5'

networks:
  default:
    external:
      name: store-front-network

services:
  reverseproxy:
    image: proxy-nginx
    container_name: proxy-nginx
    ports:
      - 80:80
      - 443:443
    restart: always

  storefront-backend:
    image: storefront-backend
    container_name: storefront-backend
    restart: always
    environment:
      - CONTAINER=storefront-backend
    ports:
      - 9082:9080
      - 9445:9443

  storefront-frontend:
    image: storefront-frontend
    container_name: storefront-frontend
    restart: always
    environment:
      - CONTAINER=storefront-frontend
    ports:
      - 9081:9080
      - 9444:9443

worker_processes 1;

events { worker_connections 1024; }

http {

  sendfile on;

  upstream storefront-backend {
    server storefront-backend:9080;
  }

  upstream storefront-frontend {
    server storefront-frontend:9080;
  }

  server {
    listen 80;

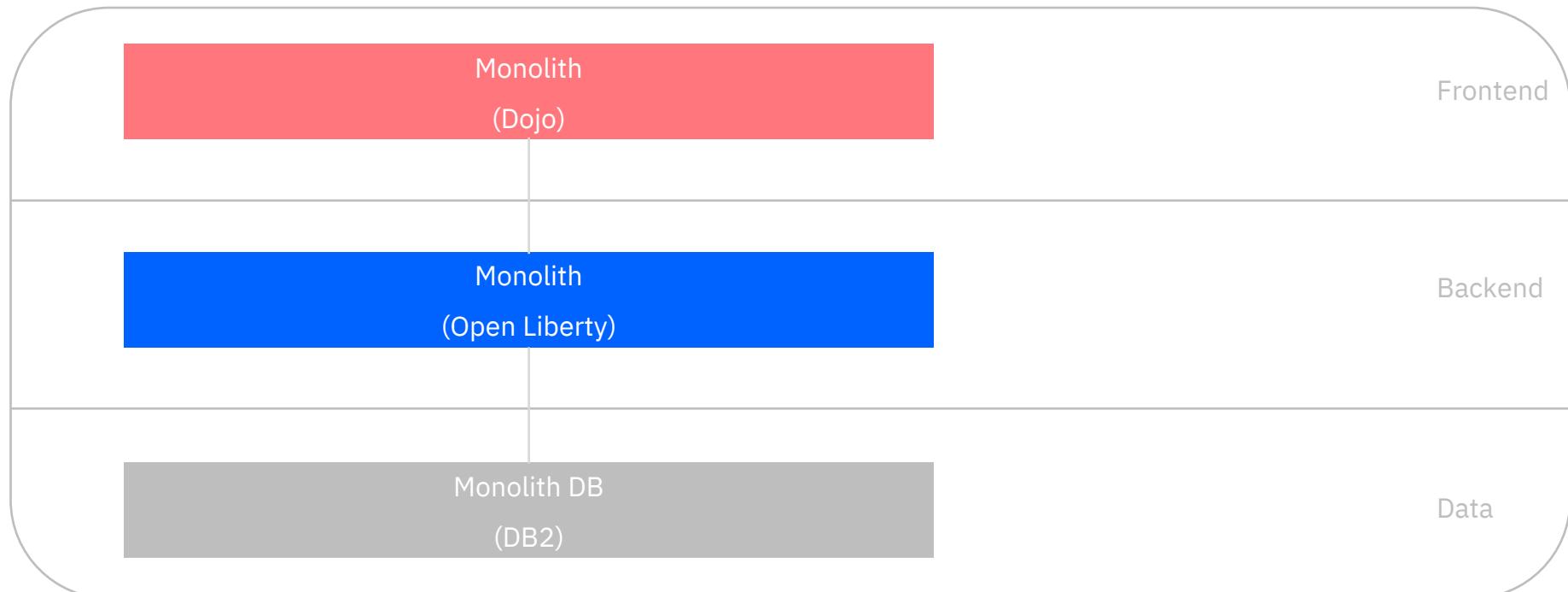
    location / {
      proxy_pass http://storefront-frontend;
      proxy_redirect off;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Host $server_name;
    }

    location /CustomerOrderServicesWeb/jaxrs/ {
      proxy_pass http://storefront-backend;
      proxy_redirect off;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Host $server_name;
    }
  }
}
```

# Step 5: Open Liberty Monolith

Description and Value	How did we get here?	Used Technologies	Containers
Usage of Open Liberty -> License cost savings	Manually (primarily changing image only)	Open Liberty MicroProfile	Backend monolith Frontend monolith Db2
-> Leverage latest MicroProfile and Open Liberty versions early and leverage open source communities			

# Step 5: Open Liberty Monolith



# Step 5: Open Liberty Monolith

## Manually created Dockerfile

```
FROM adoptopenjdk/maven-openjdk11 as BUILD
COPY . /usr/src/app/src
WORKDIR /usr/src/app/src/CustomerOrderServicesProject
RUN mvn clean package

FROM open-liberty:kernel-slim-java11-openj9
USER root
COPY --chown=1001:0 ./liberty/server.xml /config
COPY --chown=1001:0 ./liberty/server.env /config
COPY --chown=1001:0 ./liberty/jvm.options /config

ARG SSL=false
ARG HTTP_ENDPOINT=false

RUN features.sh

COPY --from=build --chown=1001:0 /usr/src/app/src/CustomerOrderServicesApp/target/CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear /config/apps/CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear
COPY --from=build --chown=1001:0 /usr/src/app/src/resources/ /opt/ol/wlp/usr/shared/resources/
RUN chown -R 1001.0 /config /opt/ol/wlp/usr/servers/defaultServer /opt/ol/wlp/usr/shared/resources &&
    chmod -R g+rwx /config /opt/ol/wlp/usr/servers/defaultServer /opt/ol/wlp/usr/shared/resources

USER 1001
RUN configure.sh
```

# Step 5: Open Liberty Monolith

## Manually created server.xml

```
<server>
  <featureManager>
    <feature>microProfile-3.3</feature>
    <feature>jakartaee-8.0</feature>
    <feature>mpReactiveStreams-1.0</feature>
    <feature>mpReactiveMessaging-1.0</feature>
    <feature>monitor-1.0</feature>
  </featureManager>

  <library id="DB2Lib">
    <fileset dir="/opt/ol/wlp/usr/shared/resources/db2" includes="db2jcc4.jar db2jcc_license_cu.jar"/>
  </library>

  <dataSource id="OrderDS" jndiName="jdbc/orders" type="javax.sql.XADataSource">
    <jdbcDriver libraryRef="DB2Lib"/>
    <properties.db2.jcc databaseName="${env.DB2_DBNAME}" password="${env.DB2_PASSWORD}" portNumber="${env.DB2_PORT}" serverName="${env.DB2_HOST}" user="${env.DB2_USER}"/>
    <connectionManager agedTimeout="0" connectionTimeout="180" maxIdleTime="1800" maxPoolSize="10" minPoolSize="1" reapTime="180"/>
  </dataSource>

  <httpEndpoint host="*" httpPort="9080" httpsPort="9443" id="defaultHttpEndpoint">
    <tcpOptions soReuseAddr="true"/>
  </httpEndpoint>

  <keyStore id="defaultKeyStore" password="whodunit"/>

  <applicationMonitor updateTrigger="mbean"/>

  <application id="customerOrderServicesApp" name="CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear" type="ear" location="CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear">
    <classloader apiTypeVisibility="spec, ibm-api, third-party" />
  </application>
</server>
```

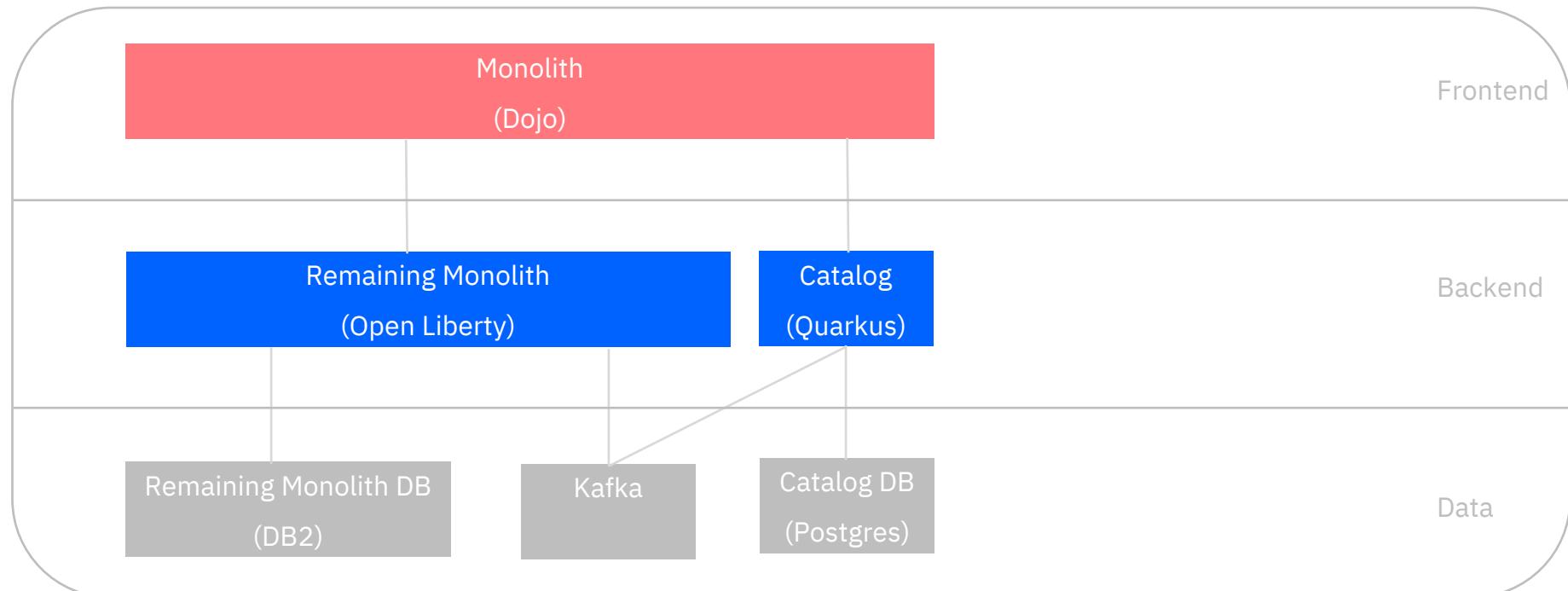
# Step 5: Open Liberty Monolith

**To be done:**  
**Document JPA issues**

# Step 6: Strangled Catalog Service and Open Liberty Monolith

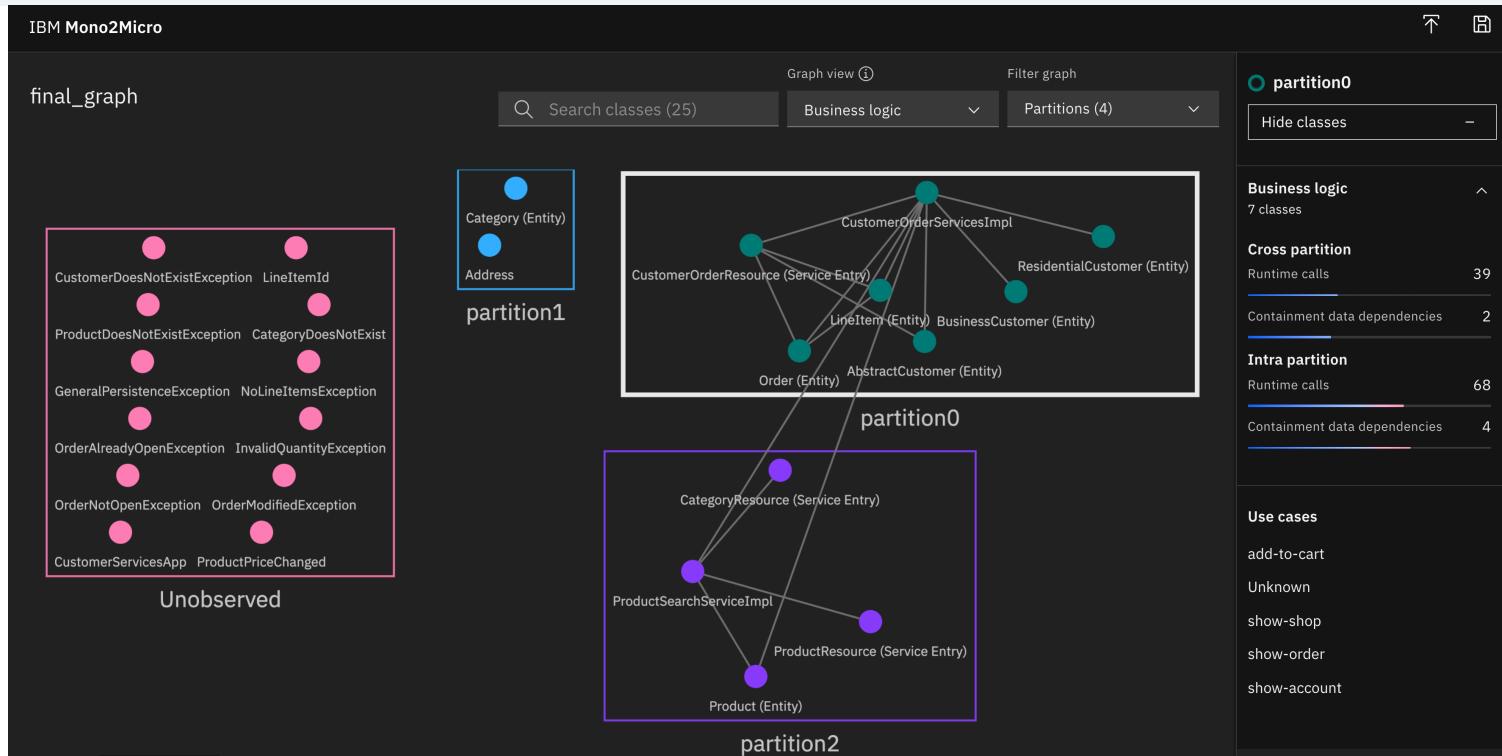
Description and Value	How did we get here?	Used Technologies	Containers
Strangled catalog service <ul style="list-style-type: none"><li>- Quarkus</li><li>- Kafka for CQRS</li><li>- Panache</li></ul>	Inspired by Mono2Micro	Quarkus OpenJ9	Backend monolith (Liberty)
Open Liberty monolith <ul style="list-style-type: none"><li>- Modern project structure</li><li>- No EJBs, but CDI</li><li>- Kafka for CQRS</li></ul>	Manually	Kafka Open Liberty MicroProfile	Catalog service Frontend monolith Db2 Kafka Postgres
-> Separate scalability			

# Step 6: Strangled Catalog Service and Open Liberty Monolith



# Step 6: Strangled Catalog Service and Open Liberty Monolith

Inspired by  
Mono2Micro



# Step 6: Strangled Catalog Service and Open Liberty Monolith

```
monolith-open-liberty
  CustomerOrderServices
    .settings
    ejbModule
    target
    .classpath
    .gitignore
    .project
    pom.xml
  CustomerOrderServicesApp
  CustomerOrderServicesProject
  CustomerOrderServicesTest
  CustomerOrderServicesWeb
    .settings
    src
    target
    WebContent
    .classpath
    .project
    pom.xml
  liberty
    jvm.options
    server.env
    server.xml
  resources
  Dockerfile
  Dockerfile.multistage
```

Remaining Open Liberty Monolith  
- new project structure:

Only one pom and one project

-> To enable Open Liberty dev tools

```
monolith-open-liberty-cloud-native
  .settings
  src/main
    java/org/pwte/example
      domain
      exception
      health
      resources
      service
      CustomerServicesApplication.java
      InventoryResource.java
      ProductPriceChanged.java
    liberty/config
      server.xml
      resources
      webapp
      target
      .classpath
      .project
      Dockerfile
      Dockerfile.multistage
      pom.xml
```

# Step 6: Strangled Catalog Service and Open Liberty Monolith

**Remaining Open Liberty Monolith**

**To be done:**

**Document how to move from EJBs to CDI**

# Step 6: Strangled Catalog Service and Open Liberty Monolith

**Remaining Open Liberty Monolith**

**To be done:**

**Document how to change JPA**

# Step 6: Strangled Catalog Service and Open Liberty Monolith

**Remaining Open Liberty Monolith**

**To be done:**

**Add health checks, logging and monitoring**

# Step 6: Strangled Catalog Service and Open Liberty Monolith

## Catalog service: Panache

```
package com.ibm.catalog;

import java.math.BigDecimal;...

@Entity
@Cacheable
@NamedNativeQuery(name="product.by.cat.or.sub",
    query="select distinct p.id,p.name,p.price,p.description,p.image from product as p,productcategory as pc, category as c where (c.id = ? or c.parent = ?) AND pc.cat
    resultClass=Product.class)
public class Product extends PanacheEntity {

    public Product() {}

    public BigDecimal price;

    public String name;

    public String description;

    public String image;

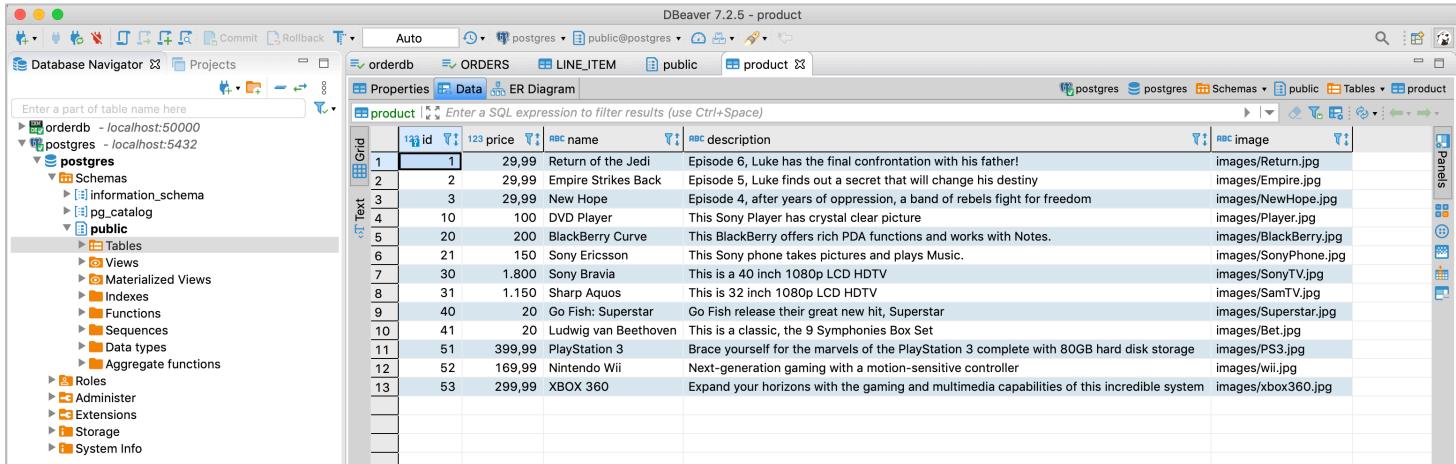
    @ManyToMany(fetch=FetchType.EAGER)
    @JoinTable(name="productcategory",joinColumns={@JoinColumn(name="productid")},inverseJoinColumns={@JoinColumn(name="categoryid")})
    public Collection<Category> categories;

    @JsonbTransient
    public Collection<Category> getCategories() {
        return categories;
    }

    @JsonbTransient
    public void setCategories(Collection<Category> categories) {
        this.categories = categories;
    }
}
```

# Step 6: Strangled Catalog Service and Open Liberty Monolith

Catalog service:  
Postgres



DBeaver 7.2.5 - product

Database Navigator

orderdb - localhost:50000  
postgres - localhost:5432

Tables

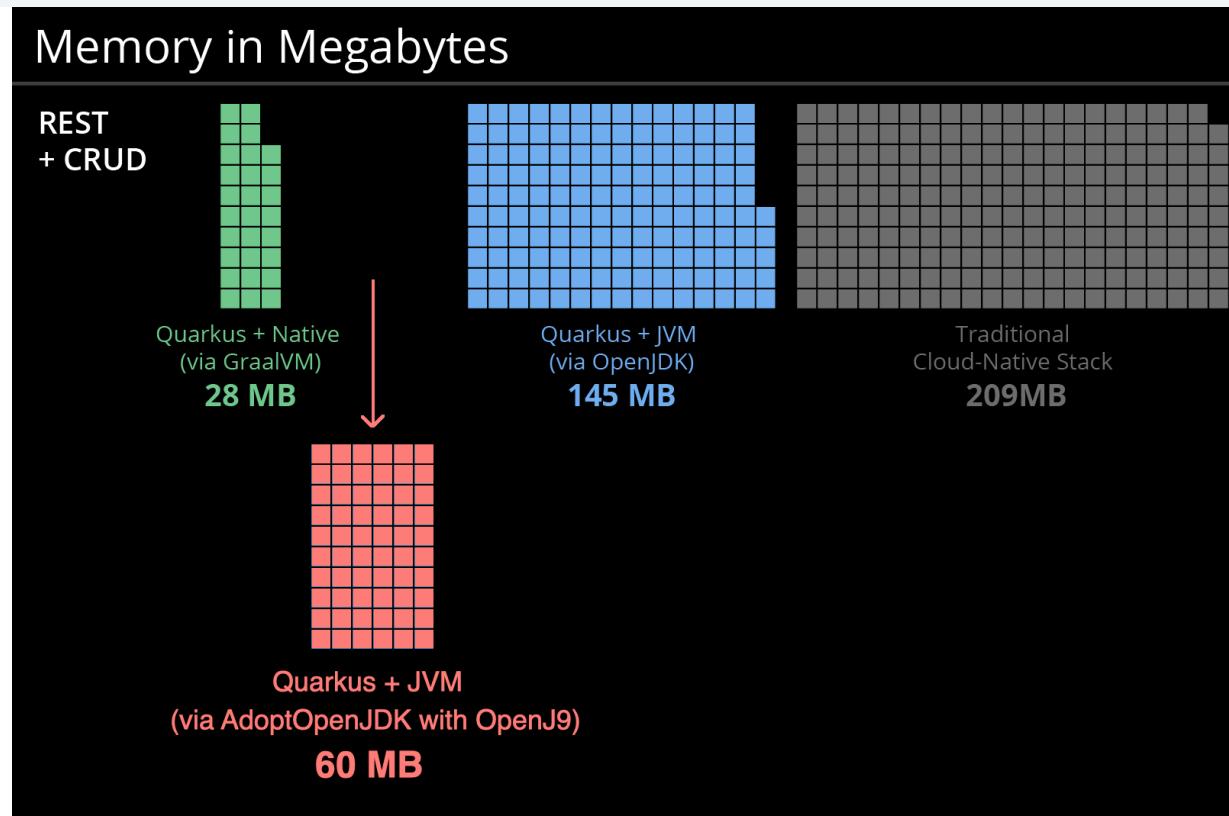
product

	id	price	name	description	image
1	1	29,99	Return of the Jedi	Episode 6, Luke has the final confrontation with his father!	images/Return.jpg
2	2	29,99	Empire Strikes Back	Episode 5, Luke finds out a secret that will change his destiny	images/Empire.jpg
3	3	29,99	New Hope	Episode 4, after years of oppression, a band of rebels fight for freedom	images/NewHope.jpg
4	10	100	DVD Player	This Sony Player has crystal clear picture	images/Player.jpg
5	20	200	BlackBerry Curve	This BlackBerry offers rich PDA functions and works with Notes.	images/BlackBerry.jpg
6	21	150	Sony Ericsson	This Sony phone takes pictures and plays Music.	images/SonyPhone.jpg
7	30	1.800	Sony Bravia	This is a 40 inch 1080p LCD HDTV	images/SonyTV.jpg
8	31	1.150	Sharp Aquos	This is 32 inch 1080p LCD HDTV	images/SamTV.jpg
9	40	20	Go Fish: Superstar	Go Fish release their great new hit, Superstar	images/Superstar.jpg
10	41	20	Ludwig van Beethoven	This is a classic, the 9 Symphonies Box Set	images/Bet.jpg
11	51	399,99	PlayStation 3	Brace yourself for the marvels of the PlayStation 3 complete with 80GB hard disk storage	images/PS3.jpg
12	52	169,99	Nintendo Wii	Next-generation gaming with a motion-sensitive controller	images/wii.jpg
13	53	299,99	XBOX 360	Expand your horizons with the gaming and multimedia capabilities of this incredible system	images/xbox360.jpg

# Step 6: Strangled Catalog Service and Open Liberty Monolith

Quarkus with OpenJ9

To be done: Update, not  
only for initial memory  
usage



# Step 6: Strangled Catalog Service and Open Liberty Monolith

Updated price from catalog service needs to be known in remaining monolith  
-> CQRS (Command Query Responsibility Segregation)

```
Niklass-MBP-2:application-modernization-javaee-quarkus nheidloff$ curl -X PUT "http://localhost/CustomerOrderServicesWeb/jaxrs/Product/1"  
-H "accept: application/json" -H "Content-Type: application/json" -d "{\"id\":1, \"price\":50}"
```

The screenshot shows a web application interface titled "Electronic and Movie Depot". At the top, there is a navigation bar with four tabs: "Shop", "Cart", "Order History", and "Account". The "Order History" tab is currently selected. Below the navigation bar is a table with the following columns: "Order Id", "Status", "Total", and "Details". A single row is visible in the table, corresponding to Order ID 1. The "Status" column shows "Current Order", the "Total" column shows "59.98", and the "Details" column contains two items: "Item:Return of the Jedi Quantity:2 Amount:29.99" and "Item:Empire Strikes Back Quantity:1 Amount:29.99". The "Current Price per Item:50" part of the "Details" text is highlighted with a red rectangle.

Order Id	Status	Total	Details
1	Current Order	59.98	Item:Return of the Jedi Quantity:2 Amount:29.99 Current Price per Item:50 Item:Empire Strikes Back Quantity:1 Amount:29.99 Current Price per Item:29.99

# Step 6: Strangled Catalog Service and Open Liberty Monolith

## Catalog service: Send event when prices change

```
@PUT
@Consumes("application/json")
@Produces("application/json")
@Path("/{id}")
@Transaction
public Product update(@PathParam("id"
System.out.println("/CustomerOrd

    Product existingProduct = entity
    if (existingProduct == null) {
        throw new WebApplicationExce
    }
    existingProduct.price = updatedP

    entityManager.persist(existingPr

    sendMessageToKafka(existingProdu

    return existingProduct;
}

private KafkaProducer<String, String> producer;

@PostConstruct
void initKafkaClient() {
    Map<String, String> config = new HashMap<>();
    config.put("bootstrap.servers", kafkaBootstrapServer);
    config.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
    config.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
    System.out.println("bootstrapping Kafka with config: " + config);

    producer = KafkaProducer.create(vertx, config);
}

public void sendMessageToKafka(Long productId, BigDecimal price) {
    String productIdString = productId.toString();
    String priceString = price.toString();
    try {
        KafkaProducerRecord<String, String> record = KafkaProducerRecord.create("product-price-updated", productIdString + "#" + priceString);
        producer.write(record, done -> System.out.println("Kafka message sent: product-price-updated - " + productIdString + "#" + priceString));
    } catch (Exception e) {
    }
}
```

# Step 6: Strangled Catalog Service and Open Liberty Monolith

Remaining monolith: New price is cached in Db2 database

The screenshot shows the DBeaver 7.2.5 interface with the 'orderdb' database selected. The 'LINE\_ITEM' table is open in the 'Data' tab. A red box highlights the 'PRICE\_CURRENT' column in the first row, which contains the value '50'. The table has columns: ORDER\_ID, PRODUCT\_ID, QUANTITY, AMOUNT, and PRICE\_CURRENT.

	123 ORDER_ID	123 PRODUCT_ID	123 QUANTITY	123 AMOUNT	123 PRICE_CURRENT
1	1	1	1	29,99	50

# Step 6: Strangled Catalog Service and Open Liberty Monolith

Remaining monolith: New price is cached in Db2 database

```
@ApplicationScoped
public class ProductPriceChanged {

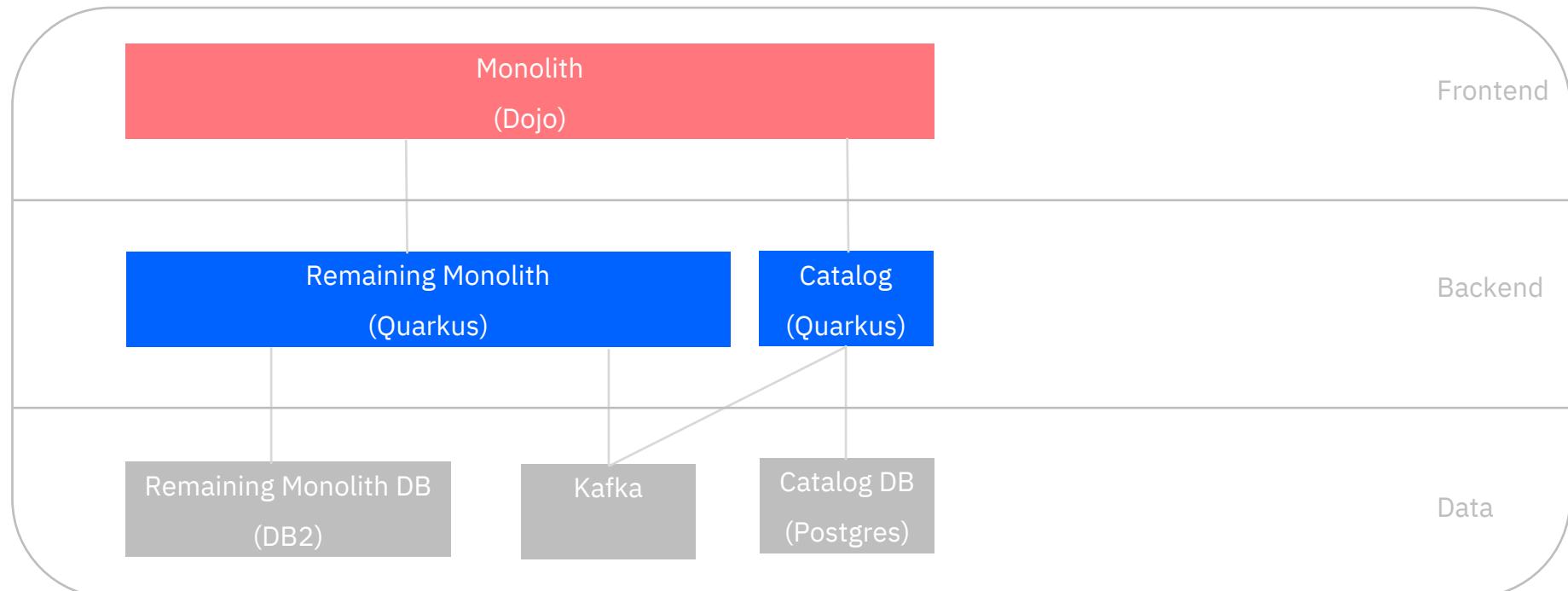
    @Inject
    CustomerOrderServicesImpl customerOrderServices;

    @Incoming("product-price-updated")
    public String process(String message) {
        System.out.println("Open Liberty - Kafka message received: product-price-updated - " + message);
        String productId = "";
        String newPrice = "0";
        try {
            productId = message.substring(0, message.indexOf("#"));
            newPrice = message.substring(message.indexOf("#") + 1, message.length());
            customerOrderServices.updateLineItem(productId, newPrice);
        }
        catch (Exception e) {}
        return message;
    }
}
```

# Step 7: Strangled Catalog Service and Quarkus Monolith

Description and Value	How did we get here?	Used Technologies	Containers
Strangled catalog service <ul style="list-style-type: none"><li>- Reactive endpoints</li><li>- Reactive Postgres access</li></ul> <p>-&gt; More efficient than synchronous code</p>	Manually	Quarkus OpenJ9 Kafka MicroProfile	Backend monolith (Quarkus) Catalog service Frontend monolith Db2 Kafka Postgres
Quarkus monolith <ul style="list-style-type: none"><li>- Compiled to native binary</li></ul> <p>-&gt; Faster startup time and less memory usage</p>			

# Step 7: Strangled Catalog Service and Quarkus Monolith



# Step 7: Strangled Catalog Service and Quarkus Monolith

Catalog service:

Reactive endpoints

```
@Path("/CustomerOrderServicesWeb/jaxrs/Product")
@ApplicationScoped
@Produces("application/json")
public class ProductResource {

    @Inject
    io.vertx.axle.pgclient.PgPool client;
    private static int MAXIMAL_DURATION = 5000;

    @GET
    public CompletionStage<List<Product>> getProductsByCategory(@QueryParam(value="categoryId") int categoryId) {
        System.out.println("/CustomerOrderServicesWeb/jaxrs/Product invoked in Quarkus reactive catalog service");

        CompletionStage<List<Product>> products = getProducts();
        CompletionStage<List<ProductCategory>> productCategories = getProductCategories();
        Long categoryIdLong = new Long(categoryId);

        return products
            .thenCombine(productCategories, (productsRead, productCategoriesRead) -> {
                List<Product> output = new ArrayList<Product>();
                List<ProductCategory> productCategoriesFiltered = productCategoriesRead.stream().filter(productCategoryRead -> {
                    return productCategoryRead.categoryid.equals(categoryIdLong);
                })
                .collect(Collectors.toList());

                productCategoriesFiltered.forEach(productCategoryFiltered -> {
                    productsRead.forEach(productRead -> {
                        if (productRead.id.equals(productCategoryFiltered.productid)) {
                            output.add(productRead);
                        }
                    });
                });
                return output;
            });
    }
}
```

# Step 7: Strangled Catalog Service and Quarkus Monolith

Catalog service:

Reactive Postgres access

```
public CompletionStage<List<Product>> getProducts() {
    String statement = "SELECT id, price, name, description, image FROM product";
    return client.preparedQuery(statement)
        .toCompletableFuture()
        .orTimeout(MAXIMAL_DURATION, TimeUnit.MILLISECONDS)
        .exceptionally(throwable -> {
            System.out.println(throwable);
            return null;
        })
        .thenApply(rows -> {
            List<Product> products = new ArrayList<>(rows.size());
            rows.forEach(row -> products.add(fromRow(row)));
            return products;
        });
}
```

# Step 7: Strangled Catalog Service and Quarkus Monolith

HTTP Request.jmx (/Users/nheidloff/Desktop/reactive/apache-jmeter-5.2.1/bin/HTTP Request.jmx) - Apache JMeter (5.2.1)

00:00:45

Test Plan

- Thread Group
  - HTTP Request
    - Summary Report – Reactive Endpoint
  - HTTP Header Manager
  - Summary Report – Reactive Endpoint
  - Response Time Graph
  - View Results Tree
  - View Results in Table

Summary Report

Name: Summary Report - Reactive Endpoint

Comments:

Write results to file / Read from file

Filename:  Browse... Log/Display Only:  Errors  Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
HTTP Req...	30000	150	5	774	70.11	0.00%	660.7/sec	46.46	101.94	72.00
TOTAL	30000	150	5	774	70.11	0.00%	660.7/sec	46.46	101.94	72.00

HTTP Request 1.jmx (/Users/nheidloff/Desktop/reactive/apache-jmeter-5.2.1/bin/HTTP Request 1.jmx) - Apache JMeter (5.2.1)

00:01:18

Test Plan

- Thread Group
  - HTTP Request
    - Summary Report – Synchronous Endpoint
  - HTTP Header Manager
  - Summary Report – Synchronous Endpoint
  - Response Time Graph
  - View Results Tree
  - View Results in Table

Summary Report

Name: Summary Report - Synchronous Endpoint

Comments:

Write results to file / Read from file

Filename:  Browse... Log/Display Only:  Errors  Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
HTTP Req...	30000	258	1	1399	83.96	0.00%	383.2/sec	760.78	59.13	2033.00
TOTAL	30000	258	1	1399	83.96	0.00%	383.2/sec	760.78	59.13	2033.00

# Step 7: Strangled Catalog Service and Quarkus Monolith

Remaining monolith (Quarkus)

Compiled to native code

```
## Stage 1 : build with maven builder image with native capabilities
FROM quay.io/quarkus/centos-quarkus-maven:20.2.0-jav11 AS build
COPY pom.xml /usr/src/app/
RUN mvn -f /usr/src/app/pom.xml -B de.qaware.maven:go-offline-maven-plugin:1.2.5:resolve-dependencies
COPY src /usr/src/app/src
USER root
RUN chown -R quarkus /usr/src/app
USER quarkus
RUN mvn -f /usr/src/app/pom.xml -Pnative clean package

## Stage 2 : create the docker final image
FROM registry.access.redhat.com/ubi8/ubi-minimal
WORKDIR /work/
COPY --from=build /usr/src/app/target/*-runner /work/application

# set up permissions for user `1001`
RUN chmod 775 /work /work/application \
&& chown -R 1001 /work \
&& chmod -R "g+rwx" /work \
&& chown -R 1001:root /work

EXPOSE 8080
USER 1001

CMD ["./application", "--quarkus.http.host=0.0.0.0"]
```

# Step 7: Strangled Catalog Service and Quarkus Monolith

Remaining monolith (Quarkus)

Quarkus doesn't allow  
blocking code on main thread

```
@ApplicationScoped
public class ProductPriceChanged {

    @PersistenceContext
    protected EntityManager em;

    @Inject
    CustomerOrderServicesImpl customerOrderServices;

    @Transactional
    @Incoming("product-price-updated")
    public CompletionStage<?> process(String message) {
        System.out.println("Quarkus - Kafka message received: product-price-updated - " + message);
        try {
            ManagedExecutor executor = ManagedExecutor.builder()
                .maxAsync(10)
                .propagated(ThreadContext.CDI,
                           ThreadContext.TRANSACTION)
                .build();

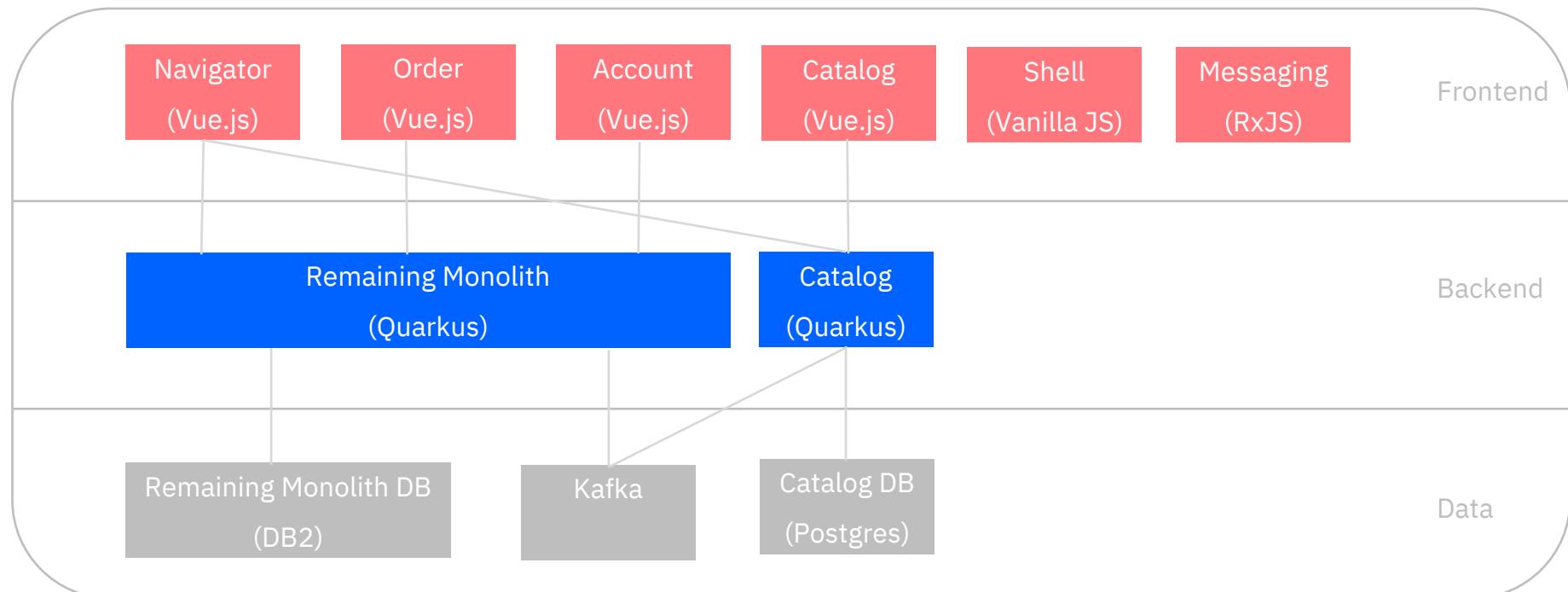
            ThreadContext threadContext = ThreadContext.builder()
                .propagated(ThreadContext.CDI,
                           ThreadContext.TRANSACTION)
                .build();

            return executor.runAsync(threadContext.contextualRunnable(() -> {
                customerOrderServices.updateLineItem(message);
            }));
        }
        catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

# Step 8: Quarkus Backend and Micro Frontends

Description and Value	How did we get here?	Used Technologies	Containers
Break down frontend monolith in micro frontends  -> Independent updates to separate pages	Manually	single-spa Vue.js RxJS	Backend monolith (Quarkus) Catalog service Frontend monolith Db2 Kafka Postgres Navigator, catalog, account, order, shell, messaging

# Step 8: Quarkus Backend and Micro Frontends



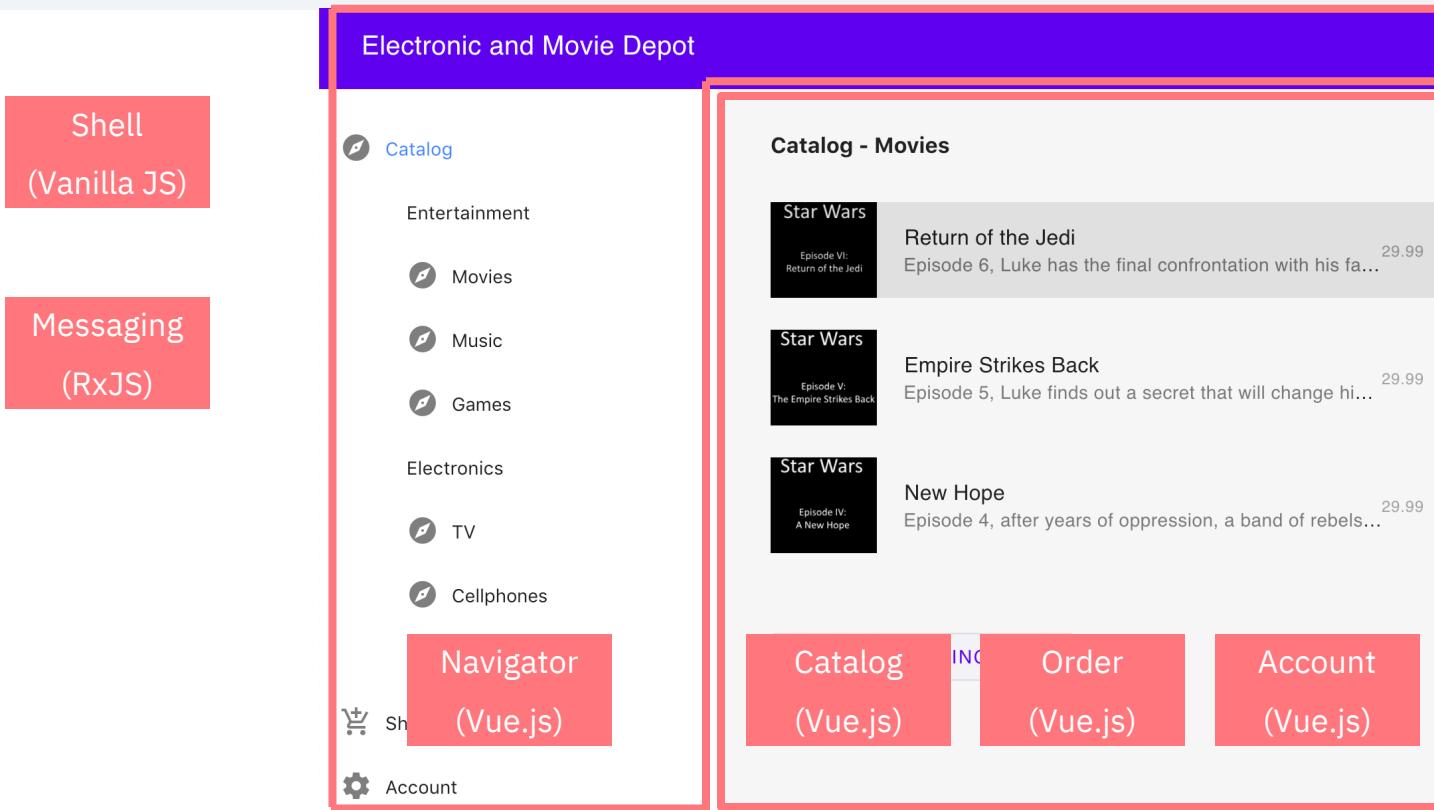
# Step 6: Strangled Catalog Service and Open Liberty Monolith

**To be done:**

**Add use case – Add ratings to products**

- **Change database**
- **Change backend code**
- **Change frontend code**

# Micro Frontends via single-spa



# Micro Frontends via single-spa

Navigator  
(Vue.js)

Catalog  
(Vue.js)

Order  
(Vue.js)

Account  
(Vue.js)

Shell  
(Vanilla JS)

```
import { registerApplication, start } from "single-spa";

registerApplication({
  name: "@vue-app-mod/navigator",
  app: () => System.import("@vue-app-mod/navigator"),
  activeWhen: ["/"],
});

registerApplication({
  name: "@vue-app-mod/order",
  app: () => System.import("@vue-app-mod/order"),
  activeWhen: ["/", "/catalog", "/order"],
});

registerApplication({
  name: "@vue-app-mod/account",
  app: () => System.import("@vue-app-mod/account"),
  activeWhen: "/account",
});

registerApplication({
  name: "@vue-app-mod/catalog",
  app: () => System.import("@vue-app-mod/catalog"),
  activeWhen: ["/", "/catalog"],
});

start();
```

Messaging  
(RxJS)

```
export default {
  ...
  getObservable(microFrontendName) {
    switch (microFrontendName) {
      case MICRO_FRONTEND_NAVIGATOR:
        if (!observableForNavigator) {
          observableForNavigator = new Subject()
        }
        return observableForNavigator
    }
  },
  send(message) {
    let topic = message.topic
    let commandId = message.commandId

    switch (topic) {
      case TOPIC_COMMAND_ADD_ITEM:
        if (observableForOrder) {
          observableForOrder.next(message)
        }
        break
    }
  }
},
```

# Micro Frontends via single-spa

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="importmap-type" content="systemjs-importmap" />
  <script type="systemjs-importmap">
    {
      "imports": {
        "single-spa": "https://cdn.jsdelivr.net/npm/single-spa@5.5.1/lib/system/single-spa.min.js",
        "vue": "https://cdn.jsdelivr.net/npm/vue@2.6.11/dist/vue.min.js",
        "vue-router": "https://cdn.jsdelivr.net/npm/vue-router@3.1.6/dist/vue-router.min.js",
        "rxjs": "http://cdnjs.cloudflare.com/ajax/libs/rxjs/6.6.3/rxjs.umd.min.js",

        "@vue-app-mod/shell": "http://localhost:8080/vue-app-mod-shell.js",
        "@vue-app-mod/messaging": "http://localhost:9001/vue-app-mod-messaging.js",
        "@vue-app-mod/navigator": "http://localhost:8501/js/app.js",
        "@vue-app-mod/order": "http://localhost:8504/js/app.js",
        "@vue-app-mod/catalog": "http://localhost:8503/js/app.js",
        "@vue-app-mod/account": "http://localhost:8502/js/app.js"
      }
    }
  </script>
  <script src="https://cdn.jsdelivr.net/npm/import-map-overrides/dist/import-map-overrides.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/systemjs/dist/system.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/systemjs/dist/extras/amd.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/systemjs/dist/extras/named-exports.js"></script>
  <link href="https://unpkg.com/material-components-web@latest/dist/material-components-web.min.css" rel="stylesheet">
  <script src="https://unpkg.com/material-components-web@latest/dist/material-components-web.min.js"></script>
  <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>
  <div style="display: flex;flex-direction: row;">
    <div id="navigator" style="order: 1;flex-grow: 2;max-width: 300px; width: 300px"></div>
    <div id="catalog" style="order: 2; flex-grow: 10;left: 320px;position: fixed;"></div>
    <div id="order" style="order: 2; flex-grow: 10;left: 320px;position: fixed;"></div>
    <div id="account" style="order: 2; flex-grow: 10;left: 320px;position: fixed;"></div>
  </div>
  <script>
    System.import('@vue-app-mod/shell');
  </script>
  <import-map-overrides-full show-when-local-storage="devtools" dev-libs></import-map-overrides-full>
</body>
</html>
```

**Catalog**

- Entertainment
- Movies
- Music
- Games
- Electronics
- TV
- Cellphones

## Catalog - Movies

Image	Title	Description	Price
Star Wars Episode VI: Return of the Jedi	Return of the Jedi	Episode 6, Luke has the final confrontation with his fa...	29.99
Star Wars Episode V: The Empire Strikes Back	Empire Strikes Back	Episode 5, Luke finds out a secret that will change hi...	29.99

**@vue-app-mod/navigator**

Default URL: <http://localhost:8501/js/app.js>

Override URL:  [\(X\)](#)

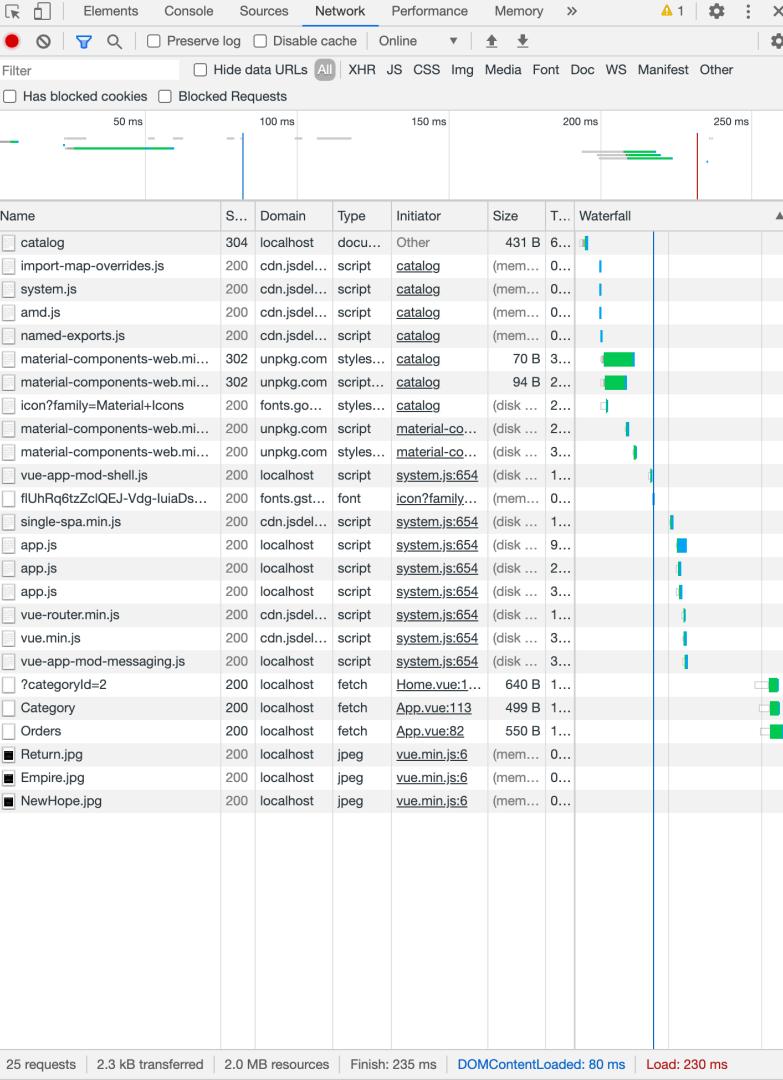
[Cancel](#) [Reset to default](#)

[ADD TO SHOPPING CART](#)

override. See documentation for more info.

Search modules [Add new module](#) [Add import map](#) [Reset all overrides](#)

Module Status	Module Name	Domain	Filename
● Inline Override	single-spa	cdn.jsdelivr.net	single-spa.dev.js
● Inline Override	vue	cdn.jsdelivr.net	vue.js
● Inline Override	vue-router	cdn.jsdelivr.net	vue-router.js
● Default	rxjs	cdnjs.cloudflare.com	rxjs.umd.min.js
● Default	@vue-app-mod/shell	localhost:8080	vue-app-mod-shell.js
● Default	@vue-app-mod/messaging	localhost:9001	vue-app-mod-messaging.js
● Default	@vue-app-mod/navigator	localhost:8501	app.js



# Micro Frontends – Messaging via RxJS for loosely Coupling Command Pattern – Example: Add Item to Cart

Catalog -> Order

```
messaging - messaging.js - send invoked - topic: TOPIC_COMMAND_ADD_ITEM commandId: messaging.js:49  
1611049822817
```

```
▶ {__ob__: we} messaging.js:50
```

```
order - App.vue - created - message: App.vue:20
```

```
Topic: TOPIC_COMMAND_ADD_ITEM App.vue:21
```

```
CommandId: 1611049822817 App.vue:22
```

```
ProductId: 1 App.vue:23
```

```
Product has been added to order App.vue:47
```

```
Products in shopping cart: 1 App.vue:54
```

```
Line items in shopping cart: 2 Order -> Navigator App.vue:55
```

```
messaging - messaging.js - send invoked - topic: messaging.js:49  
TOPIC_EVENT_AMOUNT_LINE_ITEMS_CHANGED commandId: undefined
```

```
▶ {topic: "TOPIC_EVENT_AMOUNT_LINE_ITEMS_CHANGED", payload: {...}} messaging.js:50
```

```
navigator - App.vue - amountLineItems: 2 App.vue:74
```

```
messaging - messaging.js - send invoked - topic: TOPIC_COMMAND_RESPONSE_ADD_ITEM messaging.js:49  
commandId: undefined
```

```
▶ {topic: "TOPIC_COMMAND_RESPONSE_ADD_ITEM", payload: {...}} messaging.js:50
```

```
catalog - Home.vue - message: Home.vue:69
```

Order -> Catalog