Introduction to event-driven architecture technology: Data Streaming/Kafka, CDC, Decision services, APIs, Serverless and more

October, 2019
THE EVENT
DRIVEN
ENTERPRISE
History of Personalization

<table>
<thead>
<tr>
<th>1890s - 1940s</th>
<th>1940s - 1990s</th>
<th>1990s - 2010s</th>
<th>2010s - ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demographic</td>
<td>Brand</td>
<td>Utility</td>
<td>Data</td>
</tr>
<tr>
<td>Catalogs</td>
<td>Department Stores / Malls</td>
<td>E-Commerce – Transactional</td>
<td>E-Commerce – Personalized</td>
</tr>
<tr>
<td>Limited product selection + shopping moments</td>
<td>Rising product selection + shopping moments</td>
<td>Massive product selection + 24x7 shopping moments</td>
<td>Curated product discovery + 24x7 recommendations</td>
</tr>
</tbody>
</table>

- Sears Roebuck
- Montgomery Ward
- Macy’s
- GAP
- Nike
- Amazon
- eBay
- Amazon
- Facebook
- Stitch Fix
Sense, Analyze, and Respond Cycle

- Event Driven Enterprise
- Process oriented decisions
- Reactive organization
- Macro analytics
Microservices meets typical IT

Classic Microservices Application Development

Synchronous Interaction

Asynchronous Interaction

AGILE INTEGRATION

- API
- Intermediate Store

AGILE INTEGRATION

- Industry Protocols
- ERP
- B2B
- Database
- Mainframe
- Legacy

Microservices meets typical IT

Agile Integration bridges the gap between Microservices and everything else
RISE IN EVENTS

1. MULTI-CLOUD
   Deploying applications across on-premise and public cloud drives need to sync state and notify dependent applications anywhere.

2. NEAR-REALTIME
   End users expect a near realtime experience from modern applications.

3. AVAILABILITY ISOLATION
   Deployments must be resilient by being operationally isolated for availability.

4. AGILITY
   Applications must stay highly decoupled for agility, continuous improvement, and variation.

EVENTS
ARCHITECTING FOR EVENTS
<table>
<thead>
<tr>
<th>TRADITIONAL MESSAGING</th>
<th>VS</th>
<th>EVENT STREAMING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
<td></td>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>● Store-and-forward</td>
<td></td>
<td>● long-term persistence, replay, semantic partitioning, large publisher/subscriber imbalances, replay and late-coming subscribers</td>
</tr>
<tr>
<td>● individual message exchanges (transactionality, acknowledgment, error handling/DLQs), P2P/competing consumer support</td>
<td></td>
<td>● Shared nothing data storage model</td>
</tr>
<tr>
<td>● Publish-subscribe support with limitations</td>
<td></td>
<td>● Total ordering</td>
</tr>
<tr>
<td><strong>Trade-offs</strong></td>
<td></td>
<td><strong>Trade-offs</strong></td>
</tr>
<tr>
<td>● No replay support</td>
<td></td>
<td>● Weak support for individual message acknowledgment, p2p/competing consumers</td>
</tr>
<tr>
<td>● Requires fast and/or highly available storage infrastructure</td>
<td></td>
<td>● Larger data footprint and extremely fast storage access</td>
</tr>
<tr>
<td>● No total ordering</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is Apache Kafka?

Apache Kafka is a distributed system designed for streams. It is built to be an horizontally-scalable, fault-tolerant, commit log, and allows distributed data streams and stream processing applications.
What is Kafka used for?

Use Cases

- **Messaging**
- **Web Site Activity Tracker**
- **Metrics**
- **Log Aggregation**
- **Stream Processing**
- **Data Integration**
Kafka on OpenShift with AMQ Streams

- Easy scalability
  - Running Kafka on bare metal has a high bar (ops competency, physical servers, scaling up/down, etc.)
- Automation
  - Configuration as code and automated ops via Operators
  - Tedious ops actions like rolling updates and software upgrades are greatly simplified
- High availability
  - Restoration of Kafka nodes by rescheduling pods in the event of failure
- Messaging use cases are often latency sensitive
  - Can provision cluster/topics as the same time as the application
AMQ streams (Red Hat AMQ) part of Red Hat Integration
BUILDING BLOCKS

AMQ Interconnect Router
logical federated address space
forward on best route - never store

AMQ Broker
Store-and-forward
Traditional messaging
Queuing behavior

AMQ Streams
Keep-and-serve
Streaming
Topic-heavy pubsub
Replay

OpenShift / Kubernetes
Self-service, orchestration, auto-operation, and elastic scaling

Common Protocol & APIs
Also JMS 1.1 / 2.0,
MQTT, STOMP, and
more

#redhat #rhsummit
CHANGE DATA CAPTURE AND DATA VIRTUALIZATION
Data Virtualization

- Core data federation and virtualization functions of Red Hat Data Virtualization
- Virtual databases deployed as container-native services within OpenShift
- Web-based environment for creating and managing data views
- OData access for data-driven APIs
- JDBC access for traditional clients
- Built-in integration with Fuse and 3scale for enterprise integration and API management
Change Data Capture

- Change data capture (CDC) allows database changes (inserts, updates, and deletes) to be externalized as events.
- The event stream can be used for a variety of purposes including maintaining a cache, updating search indexes, updating UIs, and generating derived views etc.
Debezium

Debezium
● Fully open-source Change Data Capture project
● Active community, led by Red Hat; see debezium.io
● Provides source connectors for popular databases
● Externalizes event stream to Apache Kafka topics

CDC in Red Hat Integration
● Debezium is being productized as part of the Red Hat Integration product
● Integrated with Apache Kafka using AMQ Streams
● Developer Preview available in Q3 release!
SERVERLESS
Serverless Defined

“computing execution model that depends on services to manage server-side logic and state where business logic run in stateless, event-triggered compute linux containers"
<table>
<thead>
<tr>
<th></th>
<th>AGILITY</th>
<th>EVENT-DRIVEN</th>
<th>FOCUS ON BUSINESS</th>
<th>OPERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>The agility of the cloud on any environment:</td>
<td>Enable event driven cloud-native applications that can also integrate with</td>
<td>Focus on business differentiation, abstract &amp; delegate infrastructure to platform</td>
<td>Consistent and scalable operations across multiple applications.</td>
</tr>
<tr>
<td></td>
<td>● On-premise</td>
<td>classic applications.</td>
<td>&amp; services.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Multi-cloud</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Hybrid</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Supersonic subatomic Java

A cohesive platform for optimized developer joy:

- Based on standards, but not limited
- Unified configuration
- Zero config, live reload in the blink of an eye
- Streamlined code for the 80% common usages, flexible for the 20%
- No hassle native executable generation
APACHE CAMEL K

- A platform for directly running integrations on Openshift and Kubernetes
- Based on Operator SDK
- Apache-based, community-driven project
- A subproject of Apache Camel started on **August 31st, 2018**

[https://github.com/apache/camel-k](https://github.com/apache/camel-k)
CASE STUDY
CHALLENGE
Swiss insurance company Helvetia faced availability and performance challenges while running its customer-facing applications on legacy, on-premise hardware. Helvetia needed to gain agility to remain competitive.

SOLUTION
Helvetia built an automated, cloud-first IT environment with greater responsiveness using Red Hat OpenShift Container Platform. The environment is enhanced by Red Hat AMQ which provides high-performance data streaming. The Red Hat AMQ streams capability integrates the features of Apache Kafka with Red Hat OpenShift Container Platform, bridging Helvetia’s legacy, mainframe infrastructure and new, modern front-end environment.

“*We wanted to move to a cloud-native software environment so we could build an engaging customer experience for new and existing applications, as well as significantly enhance agility and time to market.*”

- DR. NIKOLAS NEHMER
  Head of Helvetia Container Platform
  THE HELVETIA GROUP

**HELVETIA ACHIEVES 99.9% UPTIME FOR INSURANCE SERVICES**

| Increased Service Uptime to 99.9% | Reduced App Time-to-Market to Weeks | Improved Issue Resolution |
SUMMARY
RED HAT MIDDLEWARE APPLICATION SERVICES
SUPPORTING THE EVENT DRIVE ENTERPRISE

APPLICATION RUNTIMES
- RED HAT® OPENSHIFT® Application Runtimes
- RED HAT® DATA GRID
- RED HAT® JBOSSES APPLICATION PLATFORM
- OpenJDK
- RED HAT® AMQ BROKER

INTEGRATION
- RED HAT® FUSE
- RED HAT® AMQ
- RED HAT® 3SCALE
  API MANAGEMENT

PROCESS AUTOMATION
- RED HAT® PROCESS AUTOMATION MANAGER
- RED HAT® DECISION MANAGER

CORE TOOLS TO BUILD CLOUD
NATIVE & MIGRATE EXISTING APPS

COMPOSE AND INTEGRATE
MICROSERVICES ACROSS AN
ENTERPRISE SERVICE NETWORK

AUTOMATE AND OPTIMIZE
BUSINESS PROCESSES

Develop, deploy, and manage across cloud and on-premise

Integration with Red Hat Developer, CI/CD tools, & security services
Optimized for Red Hat OpenShift & Kubernetes services
Support organizations desire for choice and process standardization

Emphasis on solution
Simplified selling motion
Flexible consumption
Summary

- Business make better decisions with complete information in a tight “Sense - Analyze - Respond” cycle
- Predictive analytics success is predicated on real time processing of events (situational awareness)
- Change data capture, data virtualization, and a strong event processing backbone all contribute to situational awareness
- A serverless infrastructure allows your developers to focus on business logic, and results in applications that are responsive, efficient, and adaptable
Resources

- Agile Integration ebook -
- AMQ Streams overview -
- “Run Apache Kafka on Kubernetes with Red Hat AMQ streams” on demand webinar -
- Try Kafka on Kubernetes yourself! -
- Try Quarkus yourself! - https://quarkus.io/
- Try Camel K yourself! - https://github.com/apache/camel-k