

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




1890s - 1940s	1940s - 1990s	1990s - 2010s	2010s - ...
Demographic	Brand	Utility	Data
Catalogs	Department Stores / Malls	E-Commerce – Transactional	E-Commerce – Personalized
Limited product selection + shopping moments	Rising product selection + shopping moments	Massive product selection + 24x7 shopping moments	Curated product discovery + 24x7 recommendations
<ul style="list-style-type: none"> • Sears Roebuck • Montgomery Ward 	<ul style="list-style-type: none"> • Macy's • GAP • Nike 	<ul style="list-style-type: none"> • Amazon • eBay 	<ul style="list-style-type: none"> • Amazon • Facebook • Stitch Fix

Sense, Analyze, and Respond Cycle



Event Driven Enterprise

Real-time ————— Recommend products to a customer based on their shopping cart.  Subseconds to seconds

Process oriented decisions

Operational ————— Change shipment delivery schedule.  Seconds to hours

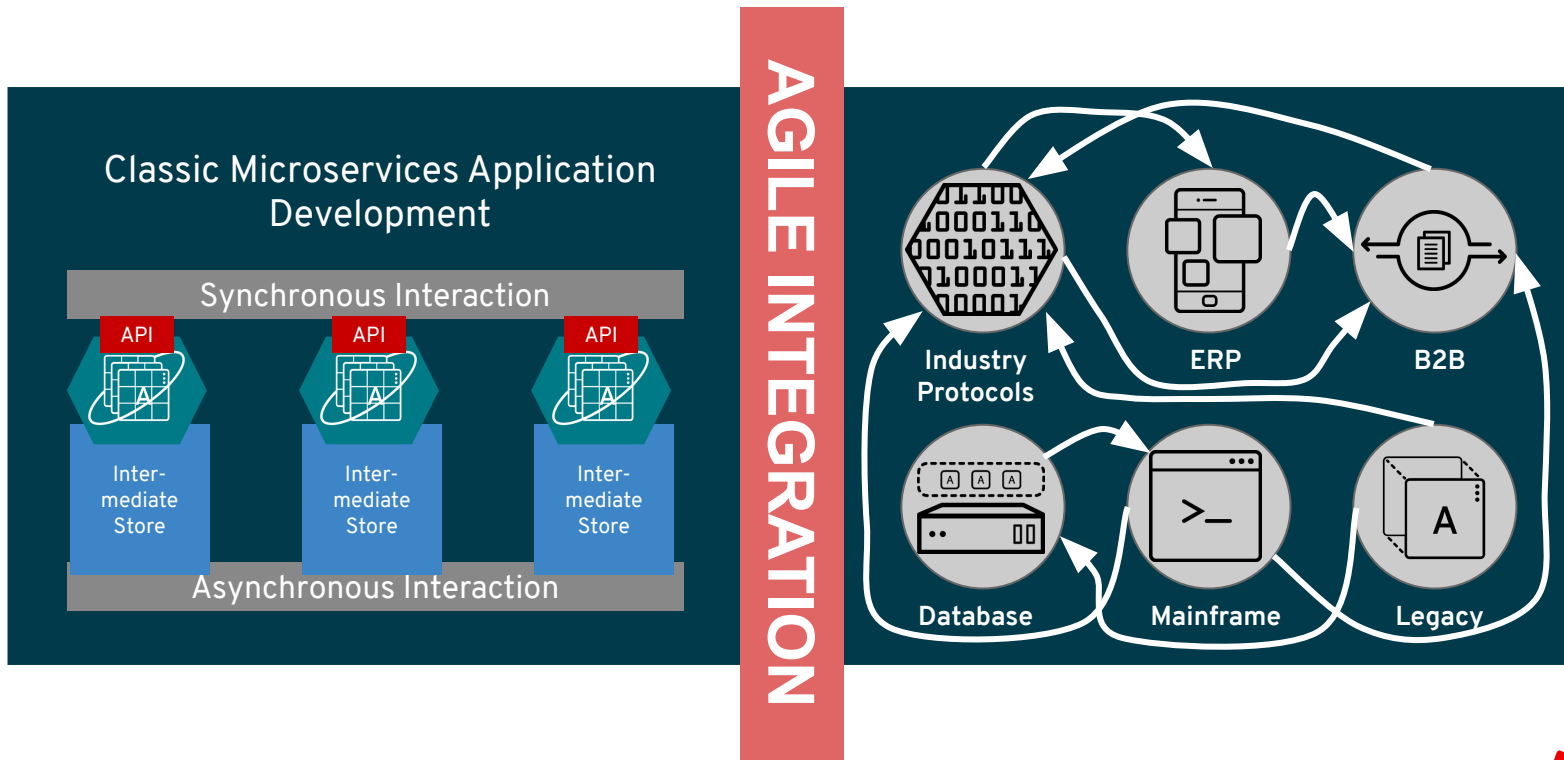
Reactive organization

Performance ————— Address product quality issues.  Hours to weeks

Macro analytics

Strategic ————— Hire more salespeople.  Weeks to years

Microservices meets typical IT



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

TRADITIONAL MESSAGING

VS

EVENT STREAMING

Advantages

- Store-and-forward
- individual message exchanges (transactionality, acknowledgment, error handling/DLQs), P2P/competing consumer support
- Publish-subscribe support with limitations

Trade-offs

- No replay support
- Requires fast and/or highly available storage infrastructure
- No total ordering

Advantages

- long-term persistence, replay, semantic partitioning, large publisher/subscriber imbalances, replay and late-coming subscribers
- Shared nothing data storage model
- Total ordering

Trade-offs

- Weak support for individual message acknowledgment, p2p/competing consumers
- Larger data footprint and extremely fast storage access

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.

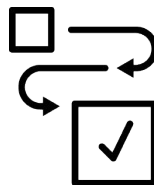


What is Kafka used for?

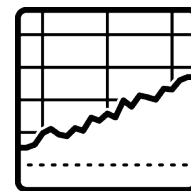
Use Cases



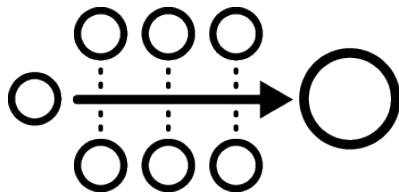
Messaging



Web Site Activity Tracker



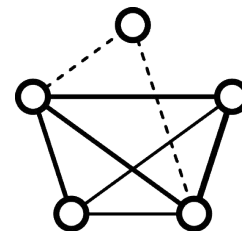
Metrics



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



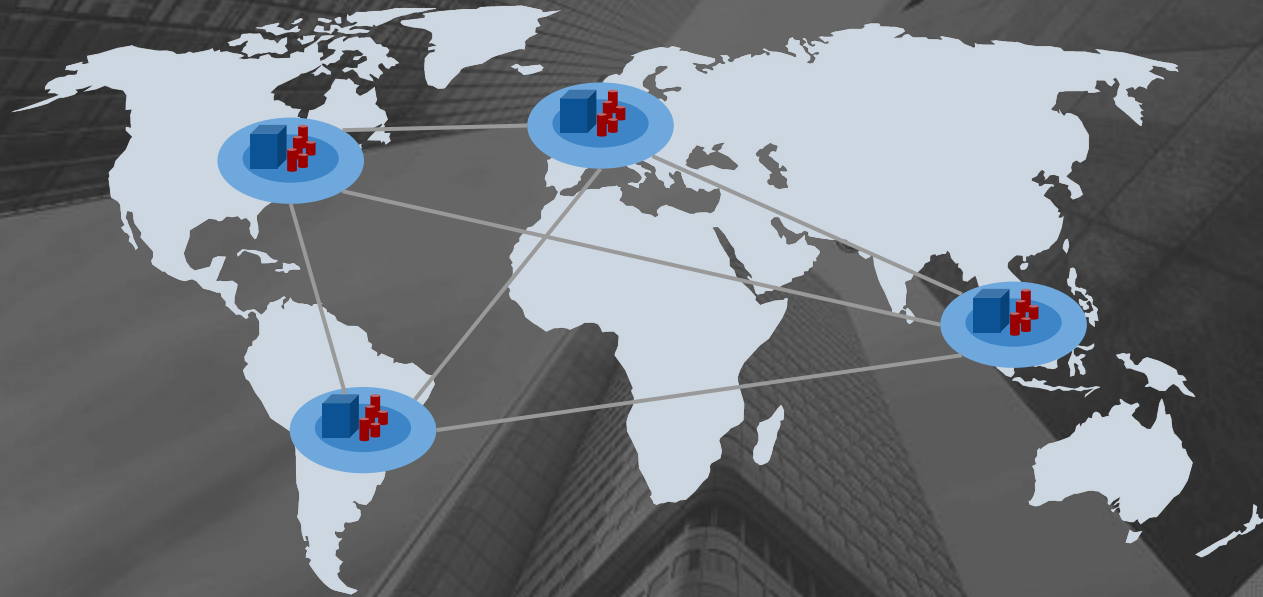
OPENSHIFT



STRIMZI



INTERCONNECT: GLOBAL NETWORK OF BANKING AND PAYMENTS SERVICES



RED HAT'S PLATFORM FOR AGILE INTEGRATION

AMQ streams (Red Hat AMQ) part of Red Hat Integration



Red Hat
Integration

RED HAT®
FUZE

RED HAT®
AMQ

RED HAT® 3SCALE®
API MANAGEMENT

BUILDING BLOCKS

AMQ Interconnect Router

logical federated address space
forward on best route - never store

AMQ Broker

Store-and-forward
Traditional messaging
Queuing behavior

AMQP 1.0

Common Protocol & APIs

Also JMS 1.1 / 2.0,
MQTT, STOMP, and
more

AMQ Streams

Keep-and-serve
Streaming
Topic-heavy pubsub
Replay

OpenShift / Kubernetes

Self-service, orchestration, auto-operation, and elastic scaling



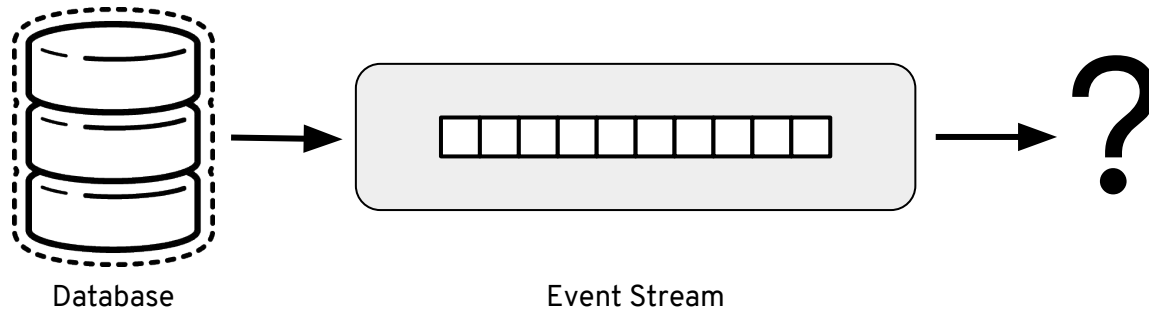
CHANGE DATA CAPTURE AND DATA VIRTUALIZATION

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!



PostgreSQL



Microsoft
SQL Server

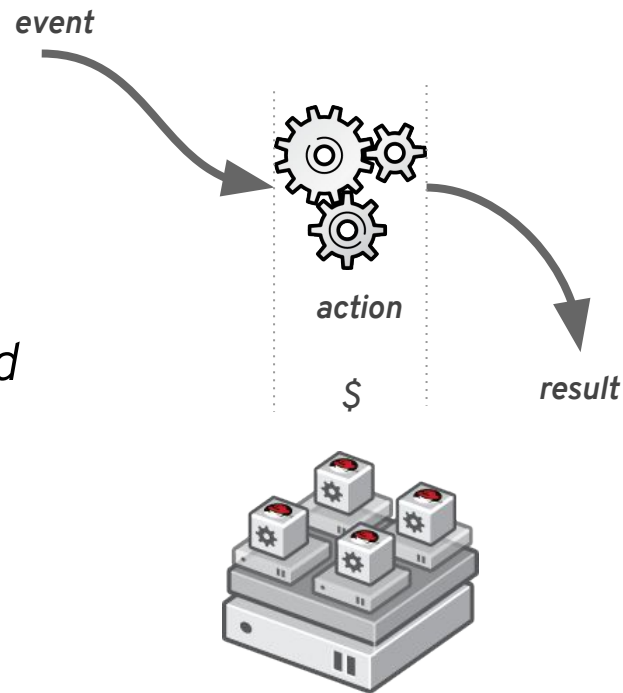




SERVERLESS

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”



Why do we need Serverless ?

01

AGILITY

The agility of the cloud on any environment:

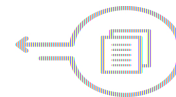
- On-premise
- Multi-cloud
- Hybrid



02

EVENT-DRIVEN

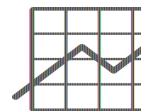
Enable event driven cloud-native applications that can also integrate with classic applications.



03

FOCUS ON BUSINESS

Focus on business differentiation, abstract & delegate infrastructure to platform & services.



04

OPERATIONS

Consistent and scalable operations across multiple applications.



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

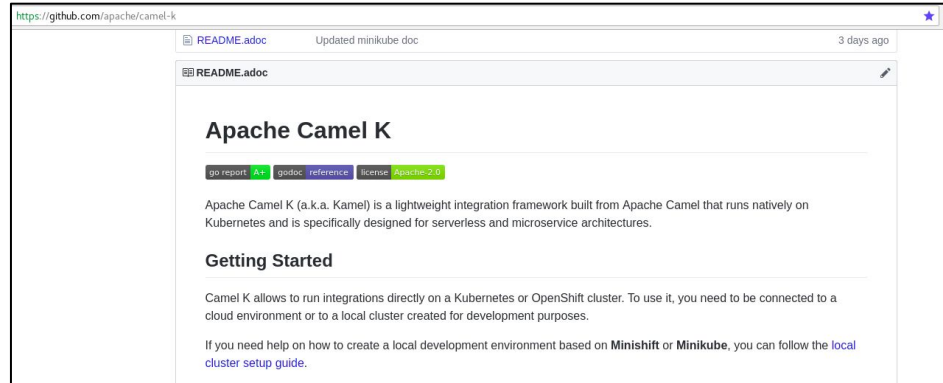
WAIT.
SO YOU JUST SAVE IT,
AND YOUR CODE IS RUNNING?
AND IT'S JAVA?!



I KNOW, RIGHT?
SUPERSONIC JAVA, FTW!



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**



CASE STUDY

HELVETIA ACHIEVES 99.9% UPTIME FOR INSURANCE SERVICES



“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

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.



Increased
Service Uptime
to 99.9%



Reduced App
Time-to-Market
to Weeks



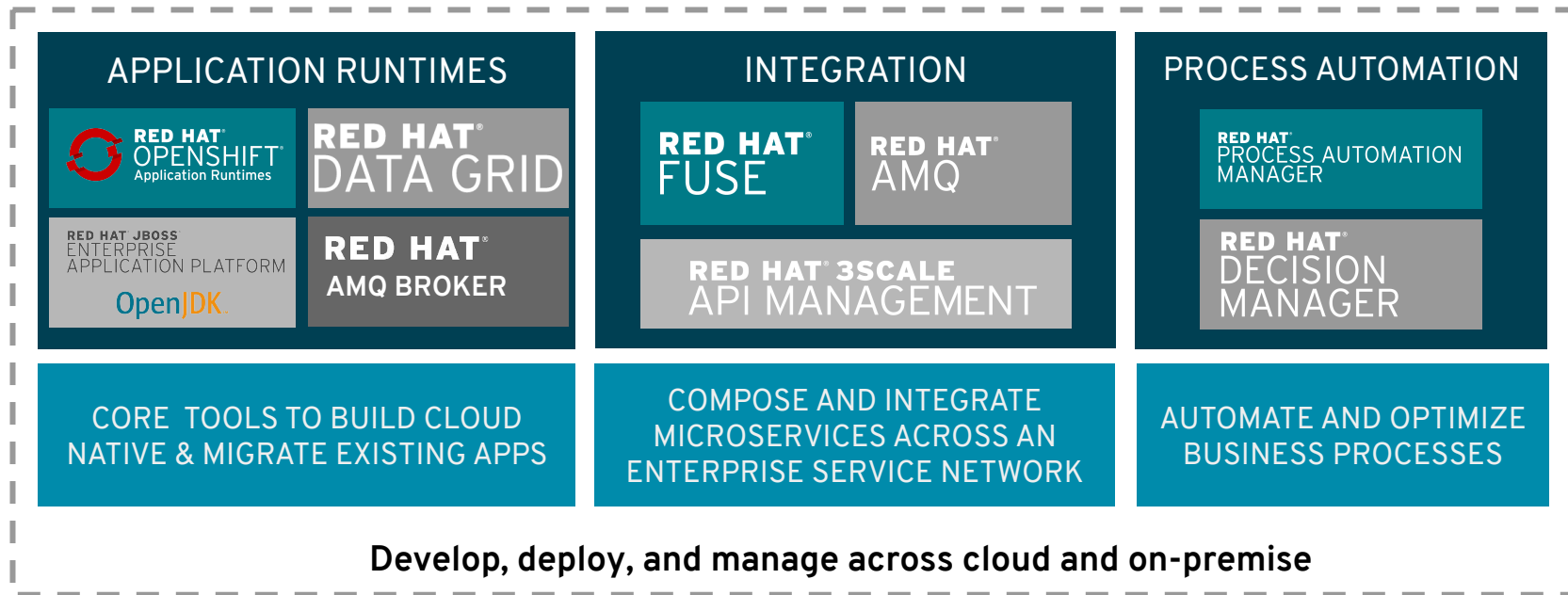
Improved Issue
Resolution



SUMMARY

RED HAT MIDDLEWARE APPLICATION SERVICES

SUPPORTING THE EVENT DRIVE ENTERPRISE



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 - <https://www.redhat.com/en/resources/mi-agile-integration-ebook>
- AMQ Streams overview - <https://www.redhat.com/en/resources/amq-streams-datasheet>
- “Run Apache Kafka on Kubernetes with Red Hat AMQ streams” on demand webinar - <https://www.redhat.com/en/events/webinar/run-apache-kafka-kubernetes-red-hat-amq-streams>
- Try Kafka on Kubernetes yourself! - <https://www.redhat.com/en/technologies/jboss-middleware/amq>
- Try Quarkus yourself! - <https://quarkus.io/>
- Try Camel K yourself! - <https://github.com/apache/camel-k>