Event-Driven Architecture

Getting started with event-driven architecture using Apache Kafka

Hugo Guerrero
APIs & Messaging Developer Advocate
Red Hat

Agenda

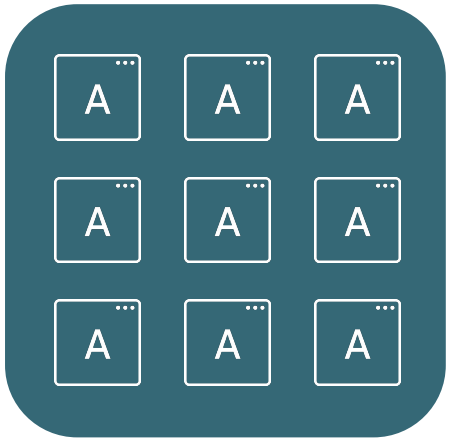
Modern App Architectures

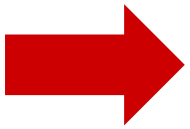
Events

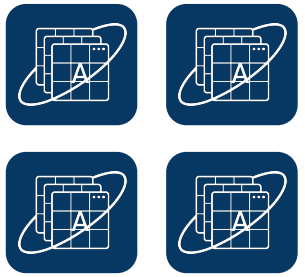
Event-Driven Microservices

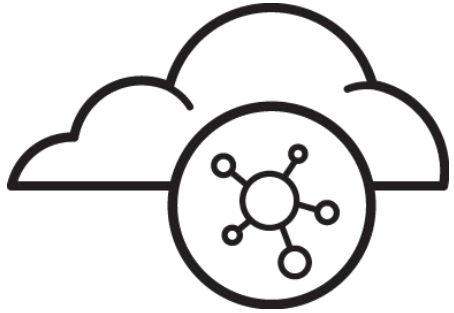
Modern Application Architectures

Modern App Architectures





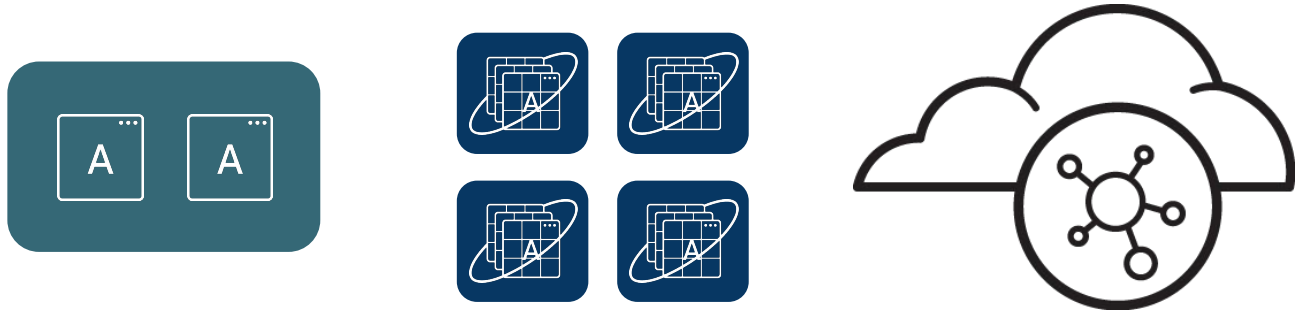




Ye Olde Architecture

Enlightened postmodernism

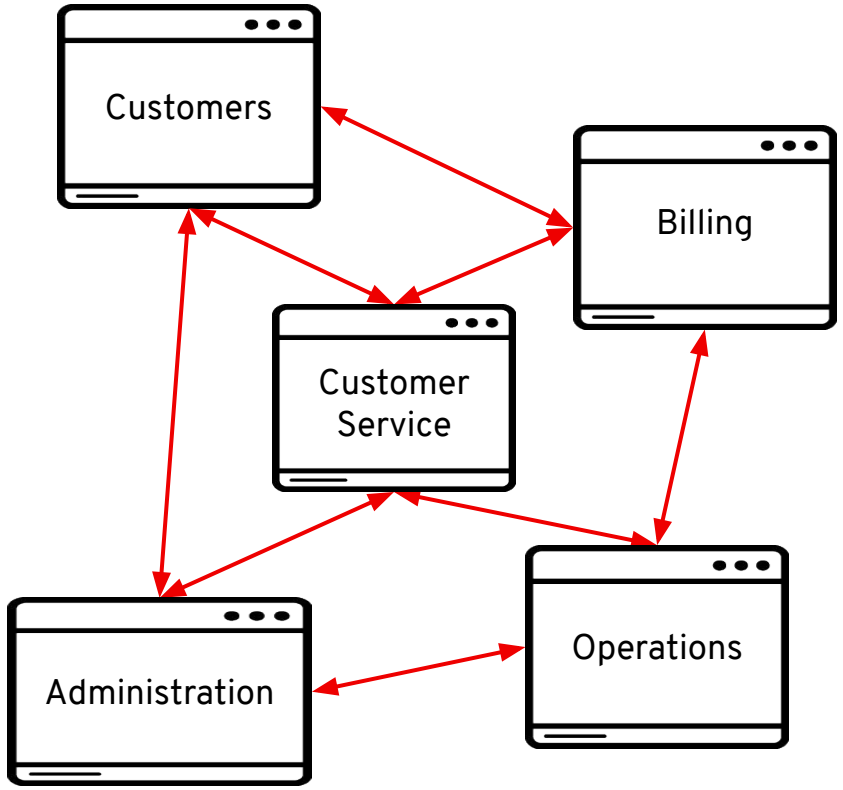
Modern App Architectures



Microservices Communications

- Distributed system
- Multiple machines
- Each service is a process
- Lightweight protocols

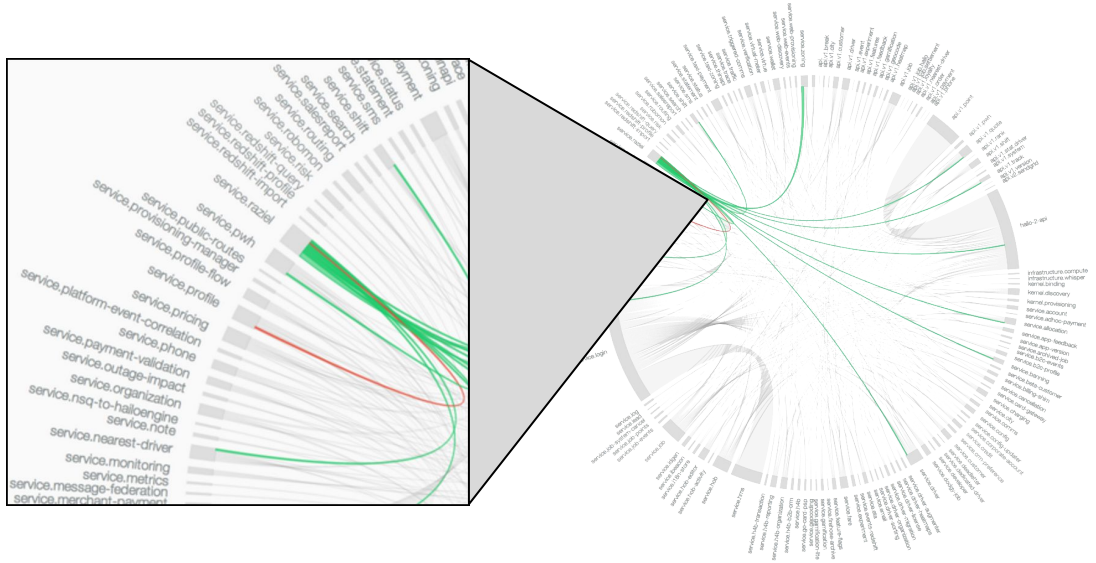
Need to talk to each other, but how?



Microservices are Hard

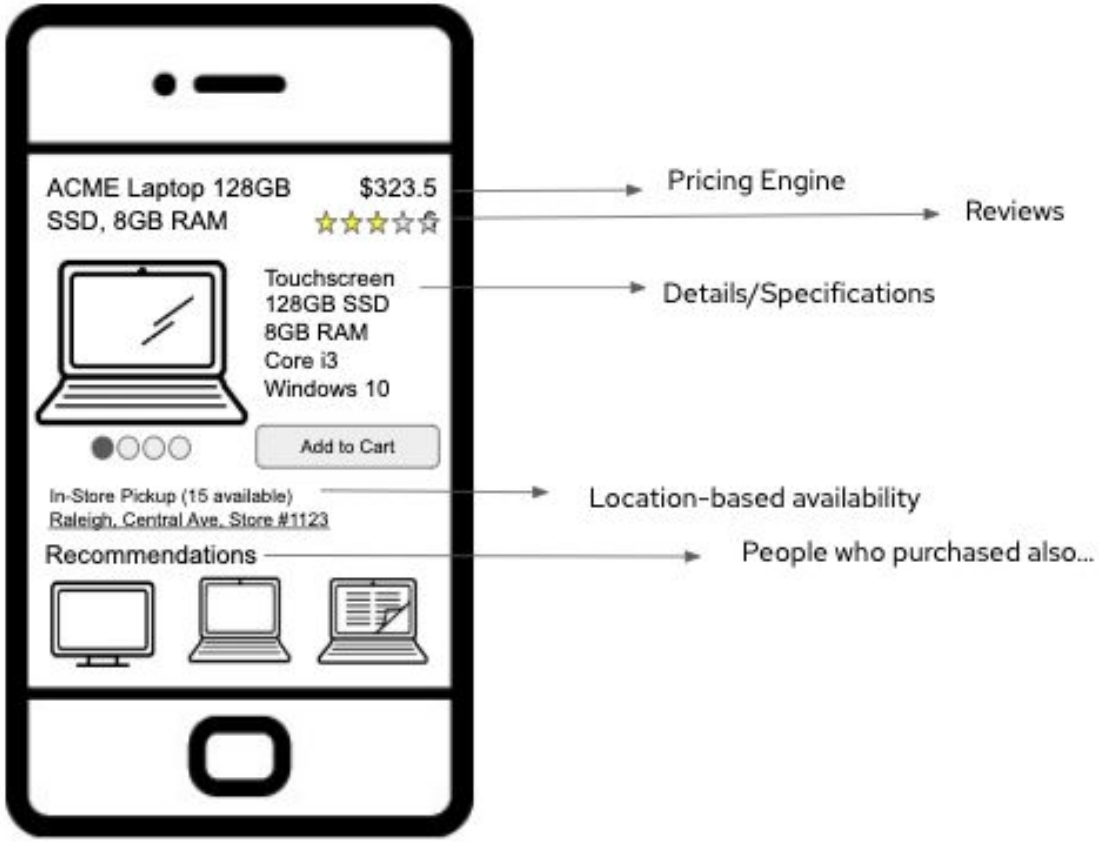
Because applications must deal with

- Unpredictable failures
- End-to-end application correctness
- System degradation
- Topology changes
- Elastic/ephemeral/transient resources
- Distributed logs
- The fallacies of distributed computing



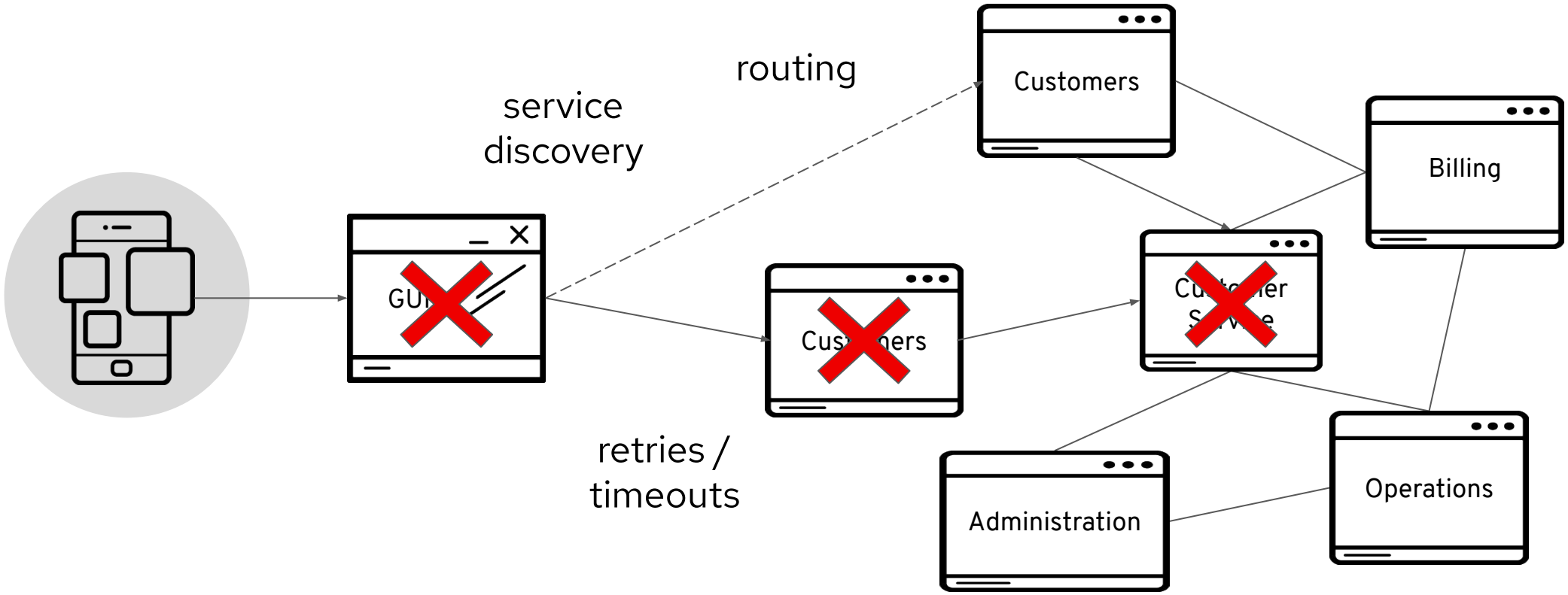
Red Hat

Example

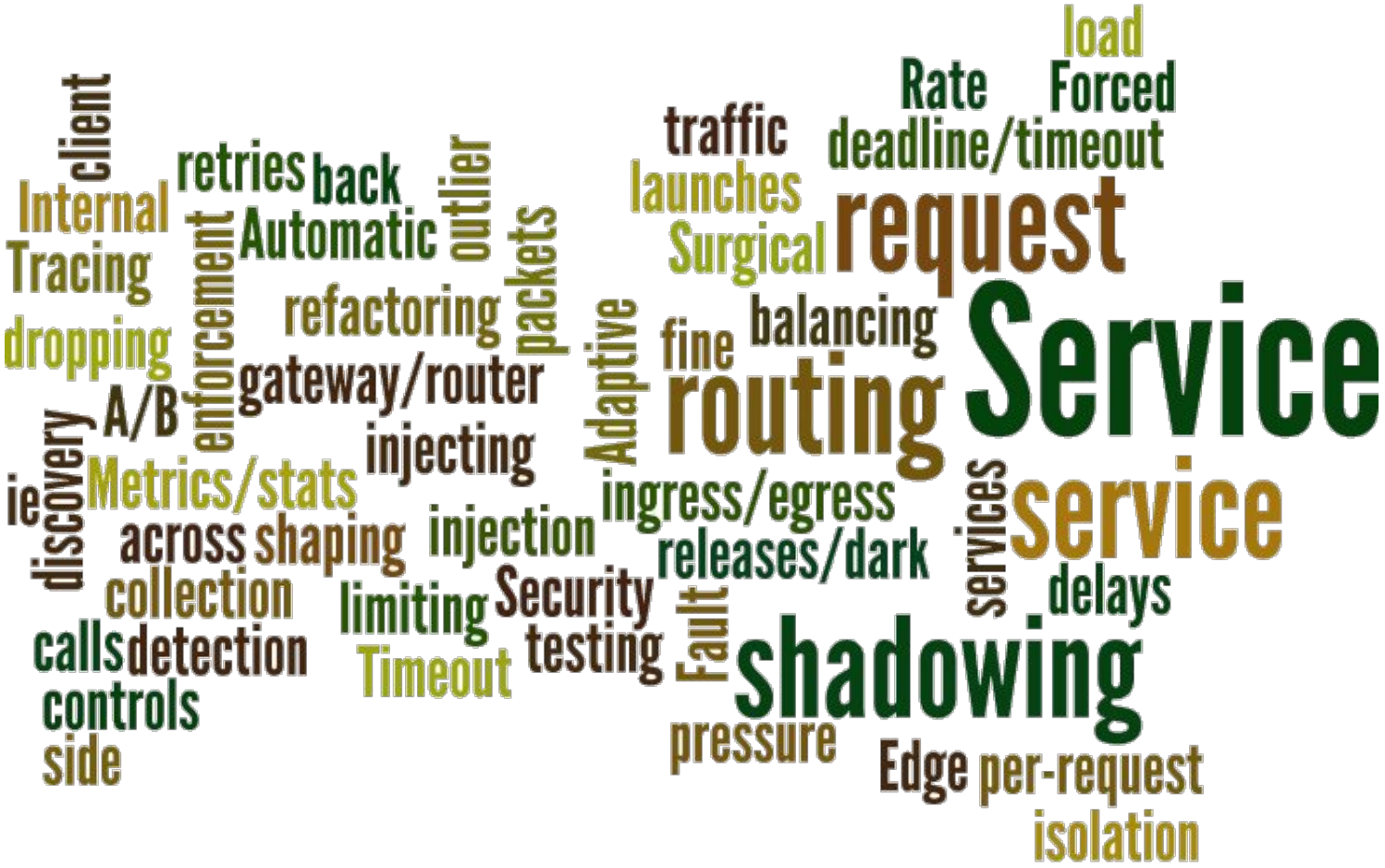


Red Hat

Handling Partial Failure



Handling Partial Failure



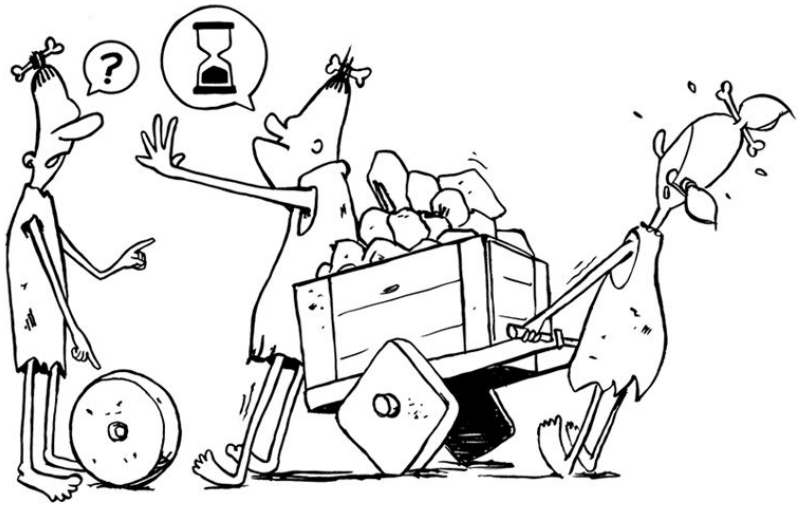
HTTP is easy but limited

- Limited fault tolerance
- No support for server initiated / peer-to-peer comms
- No help to manage transactions / long running processes
- No buffering
- No delivery guarantees

Sometimes we really need “not-so-dumb” pipes

Possible Solutions

Make developers try these



- Circuit Breaking
- Bulkheading
- Timeouts/Retries
- Service Discovery
- Load Balancing
- Traffic Control

Image source: https://www.creax.com/de/unsere-arbeit/innovation-fact-8-too-busy-to-innovate

Events

What is an Event?

Event an action or occurrence recognized by software, often originating asynchronously from the external environment, that may be handled by the software

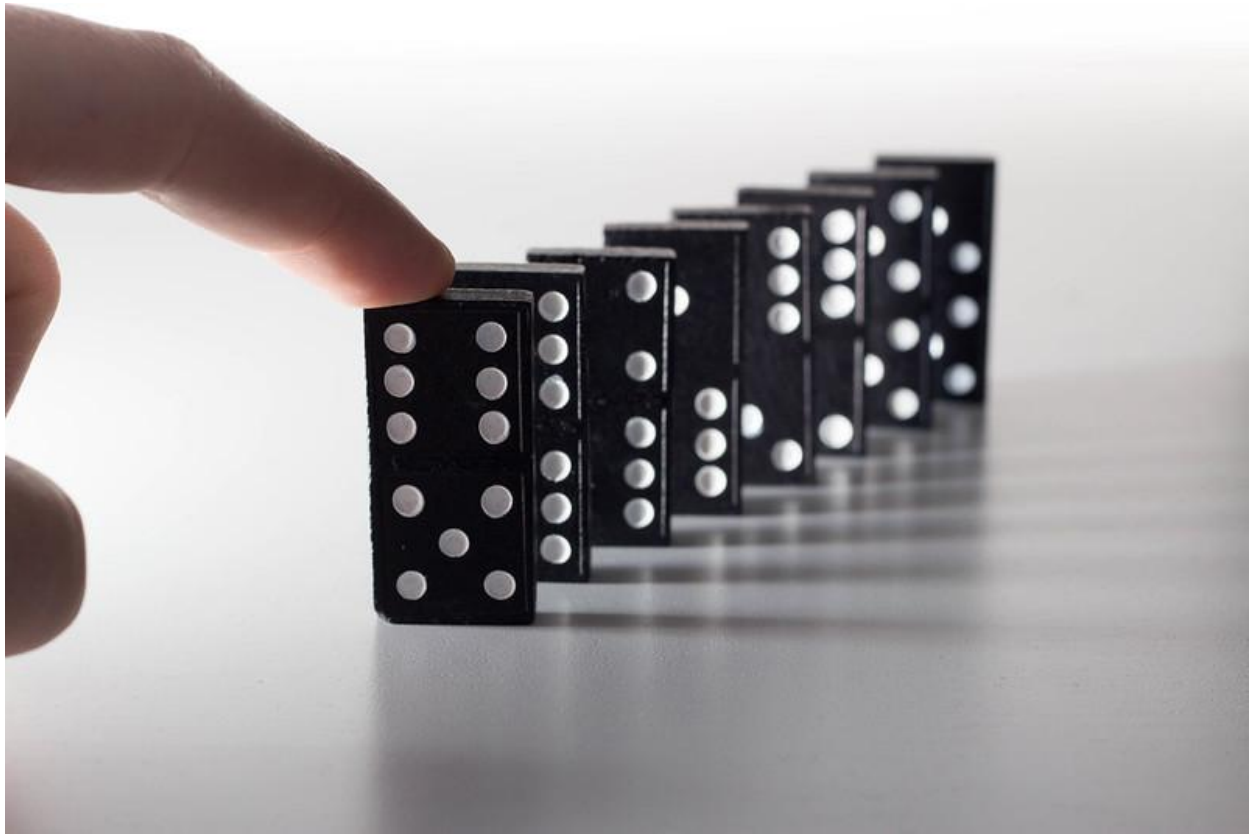
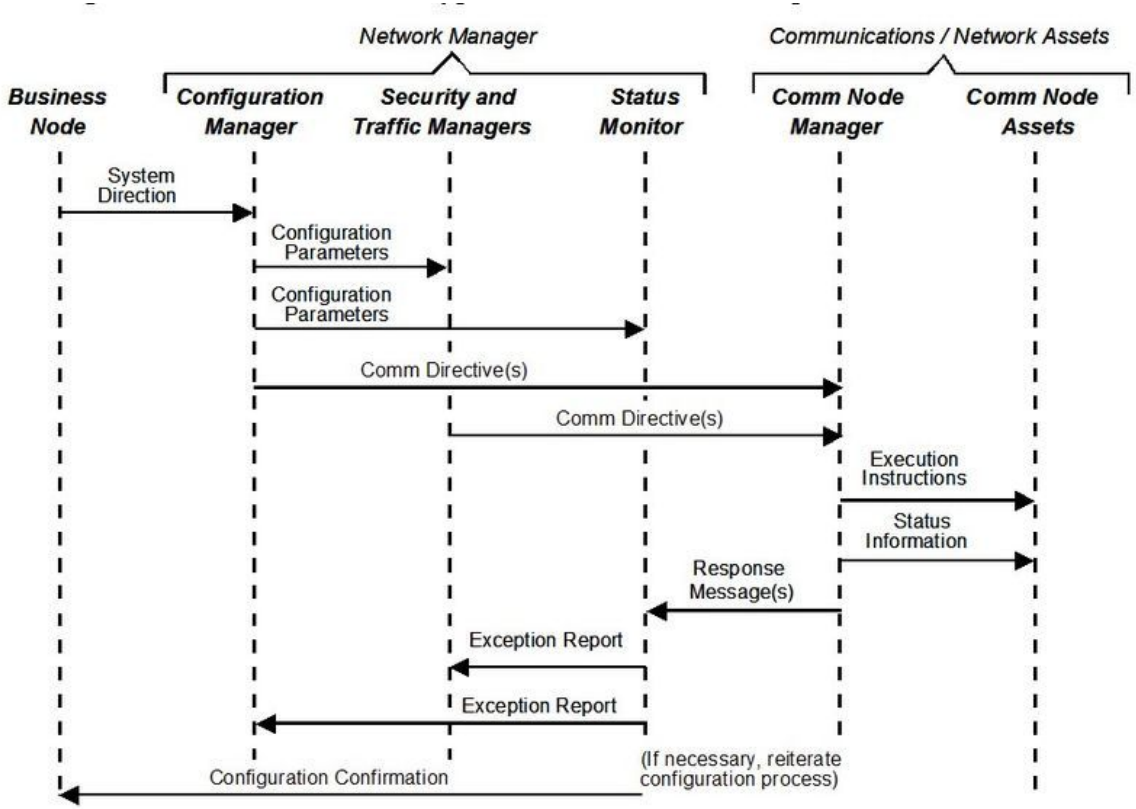


Image: https://foto.wuestenigel.com/hand-ready-to-push-domino-pieces/

What is Event Driven Architecture?

Event-Driven Architecture (EDA) is a way of designing applications and services to respond to real-time information based on the sending and receiving of information about individual event notifications



Red Hat

Why Event-Driven Architecture?

Mirrors the real world

The real world is event-driven. Systems generate and respond to events in everyday life, e.g., the human central nervous system.

Reduced coupling

Traditional RPC-style service architecture results in tightly-bound services. Changes to the application flow typically require service code changes. EDA allows new functionality to be added by adding services that consume existing event streams.

Encapsulation

Microservices concepts have grown in popularity due to the ability for service teams to develop services in isolation. EDA means that service designers need not be aware of how events are consumed.

Fine-grained scaling

Services can be independently scaled up and down to meet the event volume.

Near real-time latency

Customers increasing expect a near real-time experience. Polling on APIs is a delicate trade-off between responsiveness and load. EDA allow apps to react in near real-time without compromise.

What is an event?

Event

Immutable state and value of a particular entity, which occurred during operation among services.

Command

Async form of Remote Procedure Call, contains instructions telling recipient what to do, may cause a change of state.

Query

Similar to commands, queries expect a response returning the results, but does not cause any change in state.

Source:

Types of events

Volatile

The event needs to be disseminated to all consumers online at time of publication. Not persisted.

Consumable

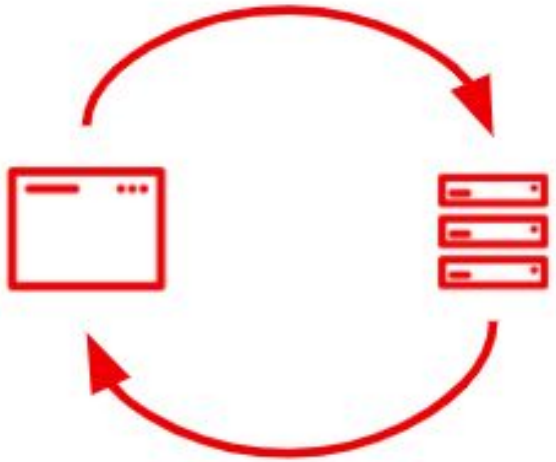
Events stored durably until read by all registered consumers. Traditional store-and-forward brokers.

Replayable

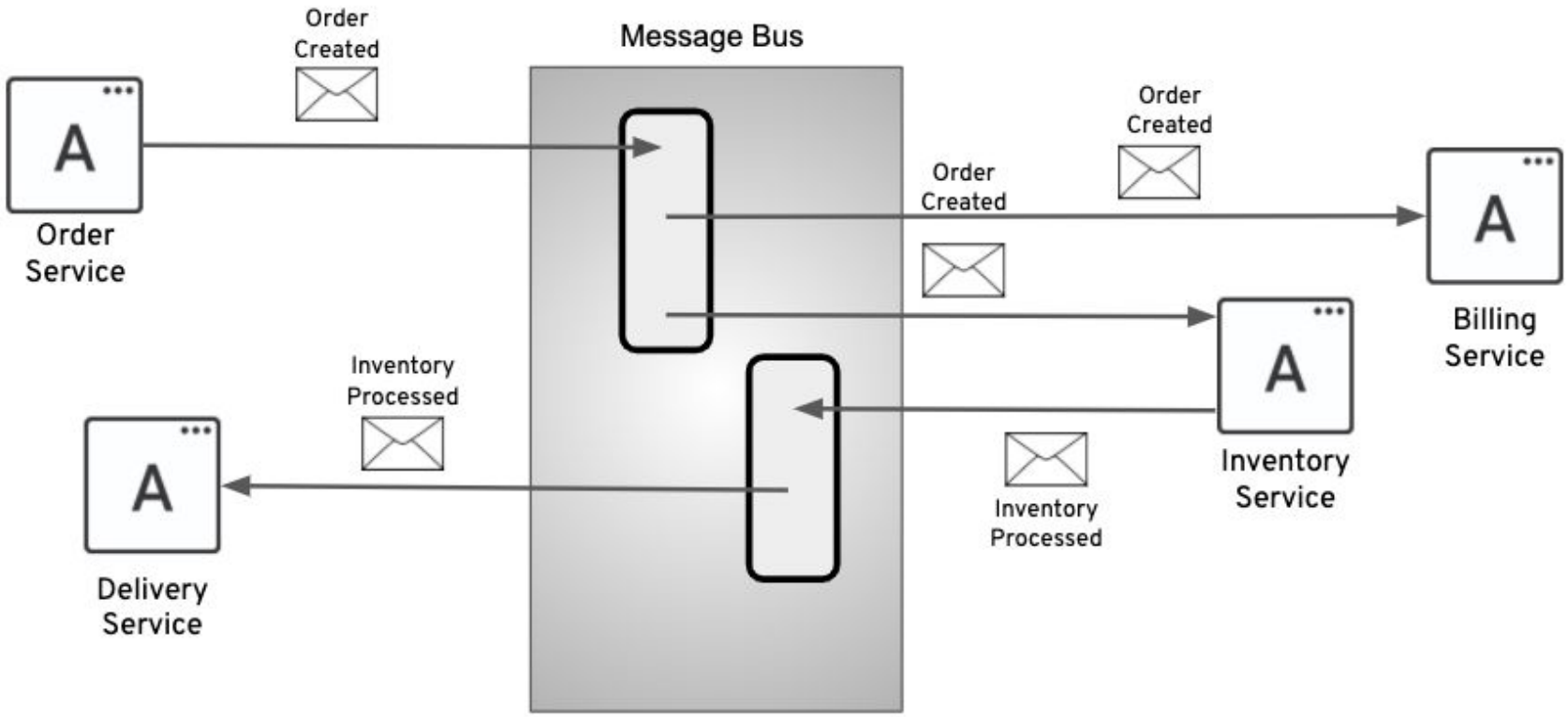
Events stored durably for specific period of time or storage capacity. Consumers can move back and forth of the stream.

Source:

Request-response / Event-driven

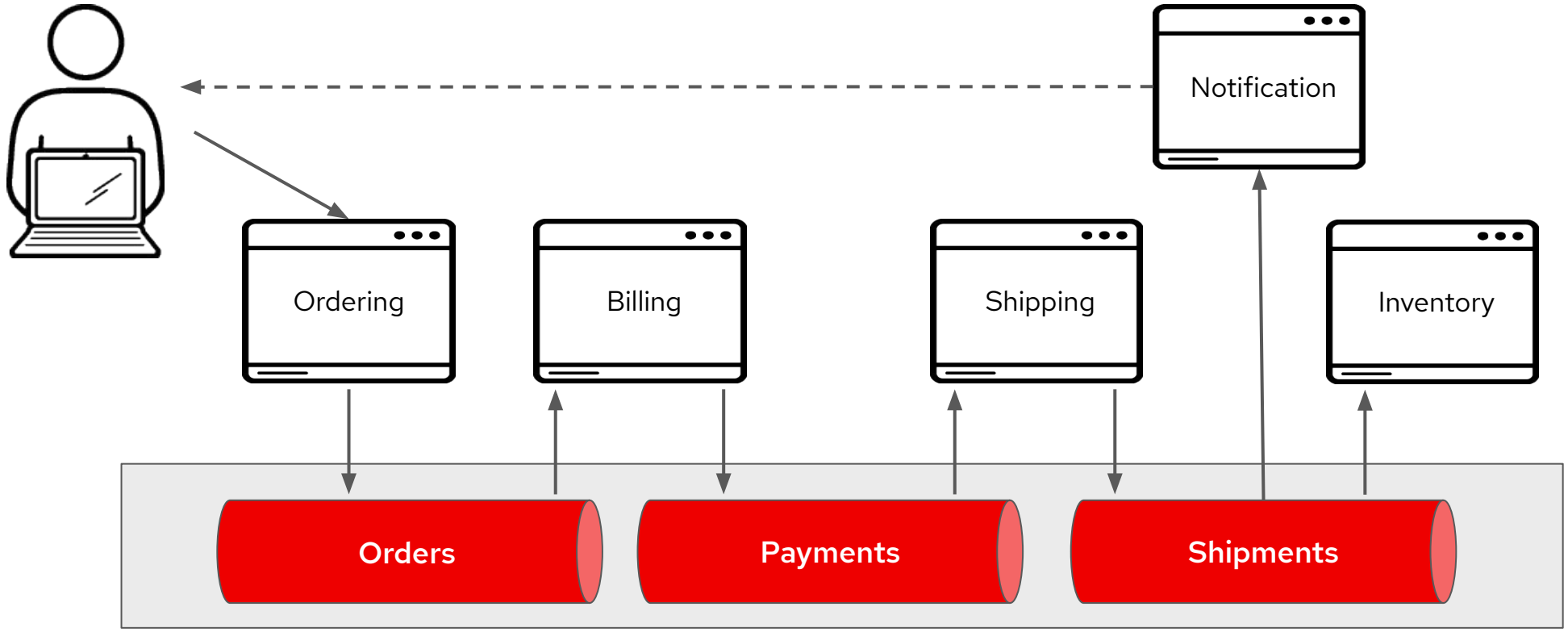


Event bus (Event Broker)



Event-driven Microservices

Event-driven Microservices

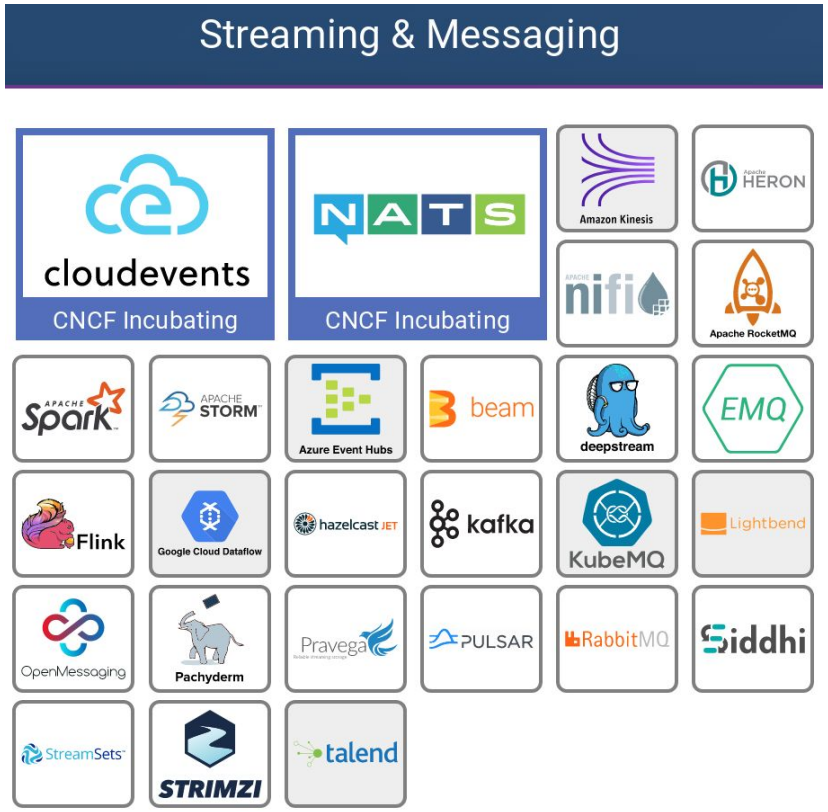


What is Apache Kafka?

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.

Kafka on Kubernetes

CLOUD NATIVE COMPUTING FOUNDATION



kafka + 

Red Hat

Running on OpenShift

- Red Hat AMQ streams provides:
  - Container images for Apache Kafka, Connect, Zookeeper and MirrorMaker
  - Kubernetes Operators for managing/configuring Apache Kafka clusters, topics and users
  - Kafka Consumer, Producer and Admin clients, Kafka Streams
- Upstream Community: Strimzi
  - 100% Open source project licensed under Apache License 2.0
  - Part of the Cloud Native Computing Foundation (CNCF)

Source:

Red Hat Integration

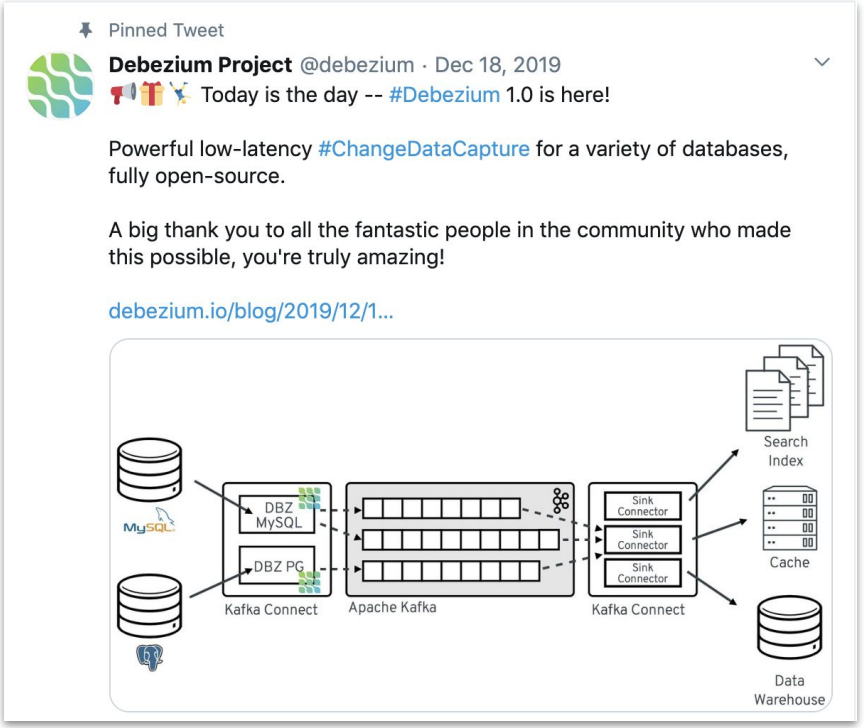
Additional components for the Apache Kafka ecosystem

- Change Data Capture Debezium Kafka Connectors
  - Streams events from your databases
- Service Registry for Red Hat Integration
  - Service and schema registry for Kafka
  - Based on Apicurio community project
- Apache Camel Kafka connectors for Kafka Connect
  - Simple usage of Apache Camel components
- Transformations through Red Hat Fuse (Kafka component)

Change Data Capture

Debezium Change Data Capture Platform

- CDC for multiple databases
  - Based on transaction logs
  - Snapshotting, Filtering etc.
- Fully open-source, very active community
- Production deployments at multiple companies (e.g. WePay, JW Player, Convoy, Trivago, OYO, BlaBlaCar etc.)



Red Hat

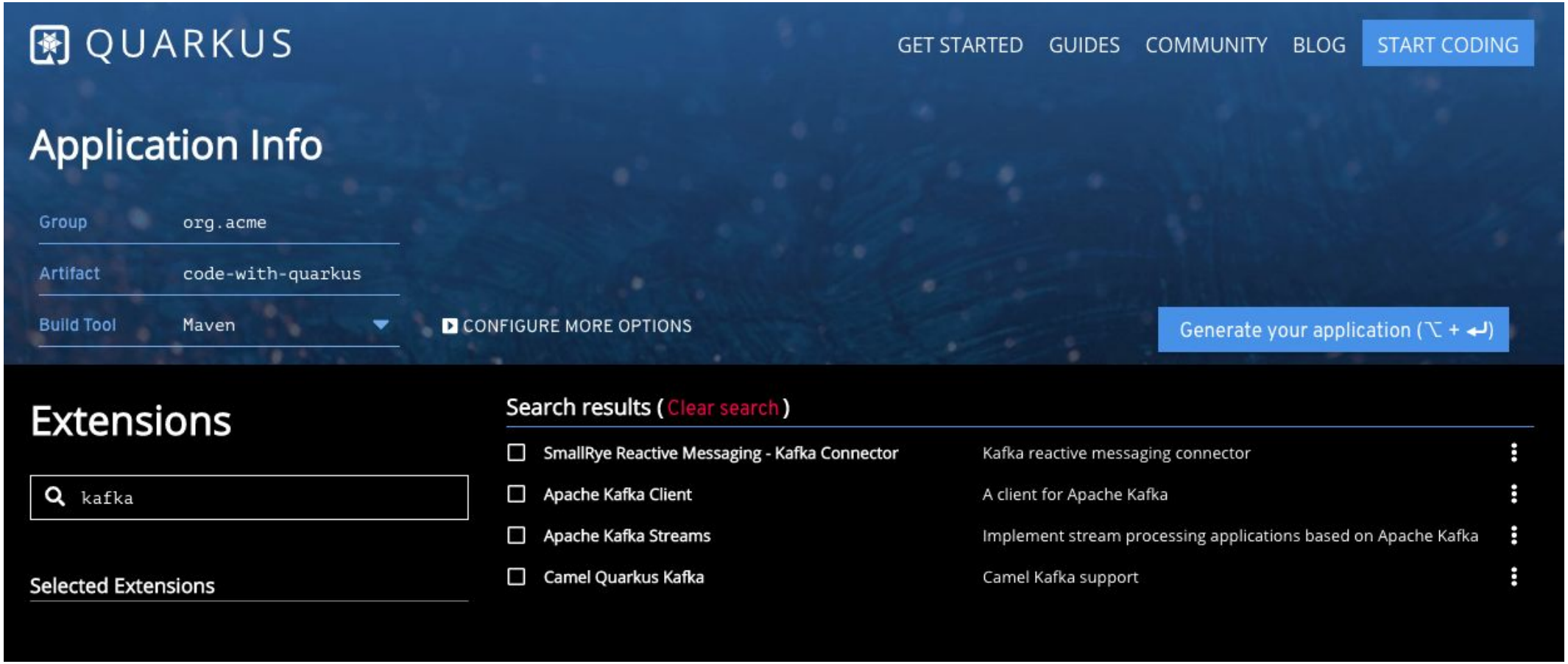
Apache Camel Kafka Connectors

- A pool of Kafka Connectors built on top of Apache Camel
- Evolved as a sub-project of the Apache Camel when donated by Red Hat to the ASF on December 2019
- Reuses in a simple way most of the Camel components as Kafka sink and sources
- Creates a (tiny) layer between Camel and Kafka Connect
- Has been conceived to expose as little of Camel as possible to appeal who comes from Kafka world
- Auto Generated documentation and connectors list

Source: https://camel.apache.org/camel-kafka-connector/latest/connectors.html

Quarkus + Apache Kafka

https://code.quarkus.io/



Source: https://code.quarkus.io/

Red Hat Integration

Implementation Patterns with Apache Kafka

Event Stream Processing

These are systems that detect and react to critical conditions by querying a continuous data stream within a small time window.

Broadcast and Pipelines

The flow of data has multiple stages and consumers, forming a pipeline, or broadcast, of information.

Event Sourcing

Event sourcing is a pattern that allows a system to log data changes in timed-order. Any service can then replay the log to determine the current data state .

Change Data Capture

The ability to capture events when data changes in the store, and auto populate these events for other services that need the latest state of the data is crucial to keep data consistent.

Source:

Event-driven Architecture and Microservices

Migration to Microservices

Microservices integration patterns that are based on events and asynchronous messaging optimize scalability and resiliency. Use message queuing services to coordinate multiple microservices, notify microservices of data changes, or as an event firehose to process IoT, social and real-time data.

Shift to Serverless

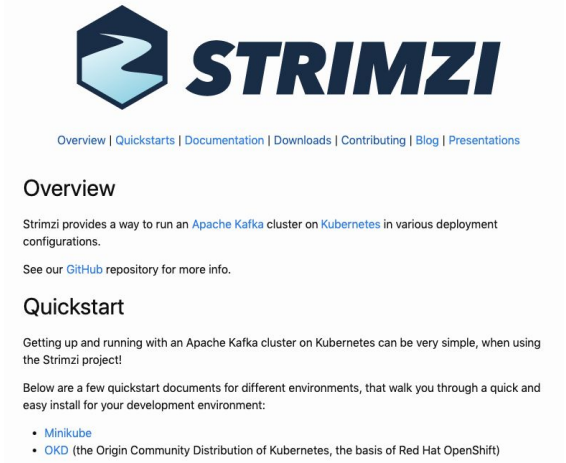
Once you've built microservices without servers, deployments onto servers, or installed software of any kind, you can use message queues to provide reliable, scalable serverless notifications, inter-process communications, and visibility of serverless functions and PaaS.

Resources

https://www.redhat.com/en/resources/event-driven-applications-amq-streams-analyst-paper

https://www.redhat.com/en/resources/mi-agile-integration-ebook

Get Started with Quarkus:
https://quarkus.io/



Get Started with Strimzi
https://strimzi.io/

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat