Event-driven Microservices

Jeremy Deane
http://jeremydeane.net
Agenda

- Event-driven Architecture (EDA)
- Micoservices Architectural Style
- Message-oriented Middleware: Apache ActiveMQ
- Enterprise Integration Patterns – Apache Camel
- Lightweight Applications – Spring Boot
- Event-driven Microservices
“All those... moments... will be lost in time, like tears... in... rain” - Roy Batty

Blade Runner
Event-driven Architecture (EDA)

**Events**

- Events represent a snapshot in time of an occurrence within a system
- Information (*noise*) to Actionable (*signal*)

**Enterprise Integration Patterns (EIP)**

- Event Message, Command Event, Event Sourcing

**Middleware Solutions**

- ActiveMQ, RabbitMQ (JMS-AMQP)
- Kafka, ZeroMQ
- Akka, Storm, Flink
EDA Principles

Events are emitted by a **Producer** and received *asynchronously*, and optionally acted upon, by a **stateless Consumer**
EDA Principles

**Events** are emitted by a **Producer** and received *asynchronously*, and optionally acted upon, by a *stateless Consumer*

**Streams** are sets of related **Events**
EDA Principles

Events are emitted by a Producer and received asynchronously, and optionally acted upon, by a stateless Consumer.

Streams are sets of related Events.

Intermediate Processors can enrich the raw Event.
EDA Principles

Ideally, Producer and Consumer should be decoupled so they can evolve independently over time. In addition, Producer should be magnanimous writer and Consumer should be tolerant reader.
EDA Principles

Ideally, Producer and Consumer should decoupled so they can evolve independently over time. In addition, Producer should be magnanimous writer and Consumer should be tolerant reader.

Consumers can listen to Event Queues or subscribe to Event Topics
EDA Examples

✧ Fraud Prevention

✧ Medical Alerting (ER Check-in)

✧ Financial Portfolio Management

✧ Supply Chain Management
“A new life awaits you in the Off-world colonies! A chance to begin again in a golden land of opportunity and adventure!” – Advertisement Blade Runner
...an approach to developing a **single application** as a suite of **small services**, each running in its **own process** and communicating with **lightweight mechanisms**, often an HTTP resource API. These services are built around **business capabilities** and **independently deployable** by fully **automated** deployment machinery. There is a bare **minimum of centralized management** of these services, which may be written in different programming languages and use different data storage technologies.

http://martinfowler.com/articles/microservices.htm

...you must be this tall to use.
- Neal Ford
Microservices Example – REST Web Services
Microservices Example – REST Web Services

Policy System
<Execution Environment>

Quote Resource
<WS Endpoint>

Renewal Resource
<WS Endpoint>

ESB
<Service Gateway>

Driver Facade
<WS Endpoint>

Care Facade
<WS Endpoint>

RMV
<SAAS>

Driver Resource
<WS Endpoint>

Carfax
<SAAS>

Car Resource
<WS Endpoint>
Microservices Example – REST Web Services

- Quote Microservice
  - Quote Resource
- Renewal Microservice
  - Renewal Resource
- Driver Microservice
  - Driver Facade
  - Driver Resource
- Car Microservice
  - Car Facade
  - Car Resource
- Carfax
- RMV

<Execution Environment>
<WS Endpoint>
<SAAS>
Breaking Down the Monolith Challenges

- Legacy Applications
- Inconsistent Domain Models
- Pandora’s *Functionality* Box
- Continuous Integration and Continuous Delivery (CI/CD) Maturity
- Organizational Structure (Conway’s Law)
Memories! You're talking about memories! - Rick Deckard Blade Runner
Apache ActiveMQ

Integration Options
Java Message Service (JMS)
Advanced Message Queuing Protocol (AMQP)

Deployment Flexibility
Stand-alone
Embedded

Advanced Topologies
Master-Slave High Availability (HA)
Federated Network

Support
Active Open Source Community
Commercial 24X7 Options

Languages - Transport
• Java-Scala – TCP/NIO
• Ruby, Perl, Python – Stomp
• C# (NMS) –TCP/NIO
ActiveMQ High Availability (HA) Messaging Backbone

failover:(tcp://PrimaryBroker:61616,tcp://SecondaryBroker:61616)?randomize=false
ActiveMQ Federated Topologies

- Network
- Business
- Other

Web Security Network Segment

Service Security Network Segment

Data Security Network Segment
Console Demo

1. Start ActiveMQ
   
   cd $ACTIVEMQ_HOME/bin
   ./activemq start OR ./activemq.bat

2. Execute Test Harness
   
   cd event-test-harness
   mvn clean install
   java -jar ./target/event-client-1.0.2.jar

3. View ActiveMQ Console
   
   Open http://localhost:8161/admin/ {admin/admin}
   Open http://localhost:8161/admin/queues.jsp

4. Start Hawtio
   
   java -jar hawtio-app.jar --port 8090

5. Open in Browser
   
   http://localhost:8090/hawtio

6. Connect → Local

7. Connect to ActiveMQ & click Agent URL
“Bryant: I need ya, Deck. This is a bad one, the worst yet. I need the old blade runner, I need your magic.” Harry Bryant Blade Runner
Domain Specific Languages (DSL) – Java, Scala, Spring DSL

Route Builders – create **Endpoints** (i.e. from & to) using **Components** (e.g. protocol) and **Processors** (e.g. Mediation, Enrichment)

**Route Engine** – Loads and executes **Routes**
/**
 * Splitter - xpath expression
 */
from("activemq:emagic.orders").
split(splitXPath).parallelProcessing().
wireTap("direct:ministry").
to("activemq:emagic.order");

/** Spring Boot – Camel Route Application */
@Configuration
@ImportResource("classpath:camel-route-spring.xml")
public class MagicRouterApplication {

    public static void main(String[] args) {
        SpringApplication.run(MagicRouterApplication.class, args);
    }
}
Apache Camel – Deployment Options

- Single Event-driven Microservice per VM
- Multiple Event-Driven Microservices per VM
- Single Event-driven Microservices Container
  - Docker, Rocket, Capsule
  - Cloud Foundry, OpenShift
Apache Camel – ActiveMQ Integration
“This announcement is brought to you by the Shimato Dominguez Corporation - helping America into the New World. – Announcement Blade Runner
Event-driven Microservices: Demo

- **Encounter Ingestion**
- **Care Management**
- **Enriched Encounter**
- **Messaging Backbone**
- **Event Auditing**
- **Audit Repository**
- **Complex Event Processing**
- **Fraudulent Encounter**
- **Queue** ingestion, audit, inpatient, cep, tasks, fraud
- **Topic** fraud
- **Queue** ingestion, audit, inpatient, cep, tasks, fraud
- **HL7 FHIR Encounter**
- **Event Auditing**
- **Audit Repository**
- **Complex Event Processing**
Event-driven Microservice - Instructions

Requires ActiveMQ listening on tcp://localhost:61616

1) Build all the sub-projects - `mvn clean install`
   - event-common
   - event-test-harness
   - event-ingestion
   - event-auditing
   - event-care-management
   - event-cep

2) Start all Microservices - `mvn spring-boot:run -Drun.arguments="-Xmx256m,-Xms128m"`
   - event-ingestion
   - event-auditing
   - event-care-management
   - event-cep
Event-driven Microservice - Instructions

Requires ActiveMQ listening on tcp://localhost:61616

3) Run Test Harness

```bash
cd event-test-harness/target
cd event-auditing/target/events
```

```bash
java -jar event-client-1.0.2.jar Leon
java -jar event-client-1.0.2.jar Roy
java -jar event-client-1.0.2.jar Eldon
```

4) View Results in ActiveMQ or Hawtio Console

```
default uid/pw = admin/admin
```

5) View Audit File Results

```
cd event-auditing/target/events
```
from("activemq:event.ingestion").
to("seda:audit");

from ("seda:audit").
  throttle(100).asyncDelayed().
  process(new TrackingIdProcessor()).
  choice().
    when().jsonpath("$[?(@.class==inpatient)]").
      to("activemq:event.inpatient").
    otherwise().
      to("activemq:event.outpatient").
  end().
to("activemq:event.audit");
from("activemq:event.ingestion").
to("direct:saas");

from("direct:saas").
  log("Sending SAAS HTTP Request").
  setHeader(Exchange.HTTP_METHOD,
    constant(org.apache.camel.component.http4.HttpMethods.GET)).
  loadBalance().
  circuitBreaker(2, 5000L, CircuitBreakerOpenException.class).
to("http4://localhost:8080/event");
“Questions... Morphology? Longevity? Incept dates?” - Roy Batty Blade Runner
Event-driven Microservices: Quality Assurance

- Source Control Management - Git
- Functional Tests - Camel Test Support
- Continuous Integration (CI) – Team City

```java
@Test
public void testSplitLogic() throws Exception {

    // Set expectations
    MockEndpoint order = getMockEndpoint("mock:order");
    order.expectedMessageCount(3);

    // read in a small batch
    String xml = IOUtils.toString(this.getClass().
                        getResourceAsStream("batch.xml"), "UTF-8");

    // execute the test
    template.sendBody("direct:split", xml);

    // should be three orders in the batch
    order.assertIsSatisfied();
}
```
Event-Driven Microservices: Operations

- Logging (e.g. Splunk, Logstash)
  - Standard Format
  - Alerts

- Processes (e.g. Zabbix, Nagios, Hyperic)
  - JMX
  - REST WS
  - Command Events (e.g. Poison Pill)

- Synthetic Transactions

- Tracking Identifier
  - Correlate activities across systems
  - Rapidly identify problems and performance bottlenecks

- Command Consoles (e.g. ActiveMQ, Java Console, Zabbix, Splunk)
Event-driven Microservices: Release Engineering

- Convention over Configuration
  - Deployment and Configuration Directory Structure
  - Linux init.d Symbolic Link

- Git Hooks
  - Trigger Release Process and Functional Regression

- Chef Recipes or AWS CloudFormation (IaaS)
  - Instantiate Linux VM
  - Provision Linux VM (Java Runtime Environment)
  - Deploy Executable JAR and Configurations

- Platform as a Service (PaaS)
  - Push Applications
“He was wrong. Tyrell had told me Rachael was special. No termination date. I didn't know how long we had together... Who does?” – Rick Deckard  

*Blade Runner*
Questions & Feedback

My Contact information:

Jeremy Deane
jeremy.deane@gmail.com
http://jeremydeane.net

https://github.com/jtdeane/event-driven-microservices
https://github.com/jtdeane/camel-standalone-router