Solving Integration Problems With JBoss Fuse

LEARN. NETWORK. EXPERIENCE OPEN SOURCE

redhat.com/summit

Scott England Sullivan - Principle Integration Architect
Peter Blinstrubas - Middleware Solutions Architect
Agenda

- Customer’s Problem
- Fuse In 0 to 60 MPH
- Architecture
- Steps to Multitenancy
- What’s Next?
- Conclusion
Customer Quote

“Camel is the right way to do integration!”
The Story

- Healthcare Company
  - Modernize SOA
  - Gain Agility
  - Reduce Cost
  - Monolith to Multitenancy
Customer’s Problem
Problem

Migrate from a COTS product while maintaining operational parity and the capability to evolve into multitenancy.
Fuse in 0 to 60 MPH
Middleware Integration

**JBoss A-MQ**

- Small-footprint messaging system
- Built for interoperability:
  - persistence, clustering, failover, security, etc

**JBoss Fuse**

- Provides Core Integration functionality
- EIP as a first-order concept
- Simplifies Integration, Transformation, Mediation

**Development with JBDS**

- Apache Active MQ
- Apache Karaf + Fuse Fabric

**Management with JON & JBoss Management Console**

- Services (Apache CXF)
- Integration (Apache Camel)
- Messaging (JBoss A-MQ)
Apache Camel

50+ Enterprise Integration Patterns

http://camel.apache.org/eip
A-MQ

- Apache Software Foundation project (ActiveMQ)
- High performance, reliable message broker
  - Clustering and Fault Tolerance
- Myriad of connectivity options
  - Native Java, C/C++, and .NET
  - AMQP, MQTT, STOMP, and OpenWire
- Embedded and standalone deployment options
Apache CXF

Standards Support

**JSR Support**

- JAX-WS - Java API for XML-Based Web Services (JAX-WS) 2.0 - JSR-224
- Web Services Metadata for the Java Platform - JSR-181
- JAX-RS - The Java API for RESTful Web Services - JSR-311
- SAAJ - SOAP with Attachments API for Java (SAAJ) - JSR-67

**WS-* & Related Specifications Support**

- Basic support: WS-I Basic Profile 1.1
- Quality of Service: WS-Reliable Messaging
- Metadata: WS-Policy, WSDL 1.1 - Web Service Definition Language
- Messaging Support: WS-Addressing, SOAP 1.1, SOAP 1.2, Message Transmission Optimization Mechanism (MTOM)
Managing Fuse with Fabric & Profiles

1 – Code artifacts are created and available in maven

2 – Fabric profiles are created that define related code to be deployed

3 – Profiles are stored in the Fabric registry

4 – Define a new container Configuration

5- Assign Profiles to the Container

6 – Start the container

Base Profile V1.0

Data Services Profile V1.0

DevContainer

Registry

runtime

DevContainer = running
Architecture
Discovery

- Auditable and Reproducible
- Scalability and Isolation
  - Grow to Multitenancy
  - Scale containers and services independently
- Standard VM environment.
- New to JBoss Fuse
Key Recommendations

- **Fuse:**
  - Validated as the best path to multitenancy
  - Encourages modular development practice
  - Fabric delivers fine to coarse grained service definitions

- **Simplification:**
  - Simplify initial physical architecture w.r.t services/Domains

- **Mutable:**
  - Sets foundation for the level of scalability and isolation required by the customer in their environment
JBOSS FUSE GATEWAY AND SERVICES DEPLOYMENT

- Producers
- Consumers

L7
F5

HTTPS Protocol (Synchronous)

The Gateway JMS Producer Transport

The asynchronous layer will decouple the incoming synchronous HTTP requests to free up resources

Network of Brokers (NoB) allows messages to move from broker to broker dependent on where the actual consumer is connected

NoB Transports

AMQ VM A
AMQ VM B
AMQ VM C

= 4 core x 4 GB VM or Bare Metal compute resource

= HTTPS Connections (Solid Line)

= JMS Connections (Dashed Line)

Fuse Gateway (GW) and Services VMs

JBoss Fuse Active

JBoss Fuse VM: All Services

Scale additional VMs when needed

The Services JMS Consumer Transport
JBoss Fuse Recommended Deployment Topology

**Fuse Gateway (GW) and Services VMs**
- JBoss Fuse VM: All Services
- JBoss Fuse VM: All Services
- JBoss Fuse VM: All Services

**Broker VMs**
- Network of Brokers
- AMQ VM A
- AMQ VM B
- AMQ VM C
- JBoss Fuse Active
- JBoss Fuse Active
- JBoss Fuse Active

**Scale additional VMs when needed**

**Fuse Fabric**
- Fabric VM
- Fabric VM
- Fabric VM

**Zookeeper Transport**
- Fabric requires an odd number of instances to reach a quorum

**+1 Instance as a standby in the case of a failure to maintain the quorum**

**VM**
- 1 core x 4 GB VM or Bare Metal compute resource

**VM**
- 4 core x 4 GB VM or Bare Metal compute resource

**HTTPS Connections (Solid Line)**

**Zookeeper Connections (Dashed Line)**
Steps to Multitenancy
STEPS TO MULTITENANCY

• First deploy monolithic application

• Identify candidates for partitioning
  - Use metrics or reliability history to identify outliers
  - Handle them on a case-by-case basis

• Consider isolating services for various forces, including:
  - Performance characteristics/contention
  - Business domain functionality
  - Deployment churn/update frequency
  - Criticality
  - Security (data at rest, in motion, etc)
  - SLAs

• Establish a CI/CD process
Fuse/Fabric – Architecture with “n” Partitions

Partition 1

- Fuse001
  - Fuse 1 Container
- Fuse002
  - Fuse 2 Container
- Fuse003
  - Fuse 3 Container

Partition 2

- Fuse004
  - Fuse 1 Container
- Fuse005
  - Fuse 2 Container
- Fuse006
  - Fuse 3 Container

Fabric Ensemble
Fuse/Fabric - Architecture “n” Partitions

Partition 1

- Fuse001
  - Fuse 1 Container
  - ESB002
    - AMQ 1 Active

- Fuse002
  - Fuse 2 Container
  - ESB002
    - AMQ 2 Active

- Fuse003
  - Fuse 3 Container
  - ESB002
    - AMQ 3 Active

Partition 2

- Fuse004
  - Fuse 4 Container
  - ESB004
    - AMQ 4 Active

- Fuse005
  - Fuse 5 Container
  - ESB005
    - AMQ 5 Active

- Fuse006
  - Fuse 6 Container
  - ESB006
    - AMQ 6 Active

Fabric Servers
What’s Next?
Growth Plan

• Process for scaling to multiple organizations
  – shared Fuse platform
  – balance central automation with autonomous teams
  – scale up to 1000s of fabric profiles
• Migration between environments
• Works within existing recommended Architecture
Next Steps

- **Today**
  - We have the foundation for multitenancy with modular services
  - Established a flexible, scalable, auditable, and repeatable architecture

- **“Tomorrow”**
  - Build out multitenancy environment
  - Establish a Dev/Ops tool chain
    - JBoss Fuse is a first class citizen in a CI/CD tool chain.
Conclusion
Conclusion

- Migration plan established
- Foundation for multitenancy
- Highly Scalable and Available
- Establish standard Dev/Ops Processes
Thank You

Any questions?
AN OPEN SOURCE CONFERENCE, BY AND FOR DEVELOPERS ACROSS THE GLOBE

June 23 - 26, 2015 | Boston, MA | USA
Solving Integration Problems With JBoss Fuse

LEARN. NETWORK. EXPERIENCE OPEN SOURCE

redhat.com/summit

Scott England Sullivan - Principle Integration Architect
Peter Blinstrubas - Middleware Solutions Architect
Agenda

- Customer's Problem
- Fuse In 0 to 60 MPH
- Architecture
- Steps to Multitenacy
- What’s Next?
- Conclusion
Customer Quote

“Camel is the right way to do integration!”
The Story

- Healthcare Company
  - Modernize SOA
  - Gain Agility
  - Reduce Cost
  - Monolith to Multitenancy
Customer’s Problem
Problem

Migrate from a COTS product while maintaining operational parity and the capability to evolve into multitenancy.

Migrate from a cots product while maintaining operational parity and the capability to evolve into multitenenancy

Establish an Integration Architecture that is scalable and provides isolation for fine grained and coarse grained services.
Fuse in 0 to 60 MPH
JBoss Fuse Architecture

Development & Tooling
- JBoss Developer Studio

Web Services Framework
- Apache CXF

Integration Framework
- Apache Camel

Asynchronous Messaging
- Apache Active-MQ

Container
- Apache karaf

Container Management Clustering & Deployment
- Fabric8

Management & Monitoring
- JBoss Operations Network
- Fuse Management Console

Development & Tooling

Development & Tooling
Middleware Integration

JBoss A-MQ

- Small-footprint messaging system
- Built for interoperability:
  - persistence, clustering, failover, security, etc

JBoss Fuse

- Provides Core Integration functionality
- EIP as a first-order concept
- Simplifies Integration, Transformation, Mediation

DEVELOPMENT WITH JBOSS WITH FUSE IDE

Management with JON & Jboss Management Console

Services (Apache CXF)
Integration (Apache Camel)
Messaging (JBoss A-MQ)

DEVELOPMENT WITH JBOSS

JBoss A-MQ
JBoss Fuse

MANAGEMENT WITH JON & FUSE MANAGEMENT CONSOLE
Apache Camel

50+ Enterprise Integration Patterns

http://camel.apache.org/eip
A-MQ

- Apache Software Foundation project (ActiveMQ)
- High performance, reliable message broker
  - Clustering and Fault Tolerance

- Myriad of connectivity options
  - Native Java, C/C++, and .NET
  - AMQP, MQTT, STOMP, and OpenWire

- Embedded and standalone deployment options

Supports publish/subscribe, point to point, message groups, out of band messaging and streaming, distributed transactions, ...

STOMP protocol enables Ruby, JS, Perl, Python, PHP, ActionScript, ...

Pre-integrated with open source integration and application frameworks

Deep integration with Spring Framework, OSGi, and Java EE
Apache CXF

Standards Support

**JSR Support**
- JAX-WS - Java API for XML-Based Web Services (JAX-WS) 2.0 - JSR-224
- Web Services Metadata for the Java Platform - JSR-181
- JAX-RS - The Java API for RESTful Web Services - JSR-311
- SAAJ - SOAP with Attachments API for Java (SAAJ) - JSR-67

**WS-* & Related Specifications Support**
- Basic support: WS-I Basic Profile 1.1
- Quality of Service: WS-Reliable Messaging
- Metadata: WS-Policy, WSDL 1.1 - Web Service Definition Language
- Messaging Support: WS-Addressing, SOAP 1.1, SOAP 1.2, Message Transmission Optimization Mechanism (MTOM)

Soap and REST stack
Managing Fuse with Fabric & Profiles

1. Code artifacts are created and available in maven

2. Fabric profiles are created that define related code to be deployed

3. Profiles are stored in the Fabric registry

4. Define a new container configuration

5. Assign Profiles to the Container

6. Start the container
Architecture
Discovery

- Auditable and Reproducible
- Scalability and Isolatation
  - Grow to Multitenancy
  - Scale containers and services independently
- Standard VM environment.
- New to JBoss Fuse
Key Recommendations

- **Fuse:**
  - Validated as the best path to multitenancy
  - Encourages modular development practice
  - Fabric delivers fine to coarse grained service definitions
- **Simplification:**
  - Simplify initial physical architecture w.r.t services/Domains
- **Mutable:**
  - Sets foundation for the level of scalability and isolation required by the customer in their environment
JBoss Fuse Gateway and Services Deployment

Diagram showing the architecture and components involved in the JBoss Fuse Gateway and Services Deployment.
JBoss Fuse Recommended Deployment Topology

[Diagram showing the recommended deployment topology for JBoss Fuse, including various components and connections.]

Legend:
- 1 core x 3GB VM - simple stub
- 2 core x 3GB VM - simple stub
- HTTP Connections (Solid Line)
- Zookender Connections (Dashed Line)
Simplified and efficient single JVM per VM
- In the absence of hard data, always start here
- Used throughout industry verticals with success

Triple Active at both GW/Services and Message Broker Layers
- Eliminates single point of failure condition if one instance is removed for maintenance

Fabric Separation of concerns
- Isolates message flows from management
- Creates independently scalable partitions
Steps to Multitenancy
**STEPS TO MULTITENANCY**

- First deploy monolith application

- Identify candidates for partitioning
  - Use metrics or reliability history to identify outliers
  - Handle them on a case-by-case basis

- Consider isolating services for various forces, including:
  - Performance characteristics/contention
  - Business domain functionality
  - Deployment churn/update frequency
  - Criticality
  - Security (data at rest, in motion, etc)
  - SLAs

- Establish a CI/CD process
Fuse/Fabric – Architecture with “n” Partitions

Partition 1
- Fuse001
  - Fuse 1 Container
- Fuse002
  - Fuse 2 Container
- Fuse003
  - Fuse 3 Container

Partition 2
- Fuse004
  - Fuse 1 Container
- Fuse005
  - Fuse 2 Container
- Fuse006
  - Fuse 3 Container

Fabric Ensemble
Key take away is Jboss fuse provides the flexibility to architect a solution that meets the needs of your organization. Not everyone else.
What's Next?
Growth Plan

• Process for scaling to multiple organizations
  – shared Fuse platform
  – balance central automation with autonomous teams
  – scale up to 1000s of fabric profiles
• Migration between environments
• Works within existing recommended Architecture
Next Steps

• Today
  – We have the foundation for multitenancy with modular services
  – Established a flexible, scalable, auditable, and repeatable architecture

• “Tomorrow”
  – Build out multitenancy environment
  – Establish a Dev/Ops tool chain
    • JBoss Fuse is a first class citizen in a CI/CD tool chain.

JBoss Fuse is a first class citizen in a CI/CD tool chain.

• Take advantage of Fuse’s native CI/CD capabilities
• CI/CD process includes automated tests/integration/acceptance, etc) to enact change to a production system.
• Increases visibility into the auditable and reproducible environment.
• Establish operational metrics feedback loop.
Conclusion
Conclusion

- Migration plan established
- Foundation for multitenancy
- Highly Scalable and Available
- Establish standard Dev/Ops Processes
Thank You
Any questions?
AN OPEN SOURCE CONFERENCE, 
BY AND FOR DEVELOPERS ACROSS THE GLOBE 
June 23 - 26, 2015 | Boston, MA | USA