



Fuse Technology Presentation

Charles Moulliard (@cmoulliard)

Agenda

- What is Fuse Technology
- Integration Platform
- Routing & Mediation : Apache Camel, CXF
- Modular Container : Apache Karaf
- Management & Provisioning : Fabric8, Hawtio
- Security Policy Management : ApiMan, Keycloak
- Demo

Who



Principal Solution Architect & Fuse Expert

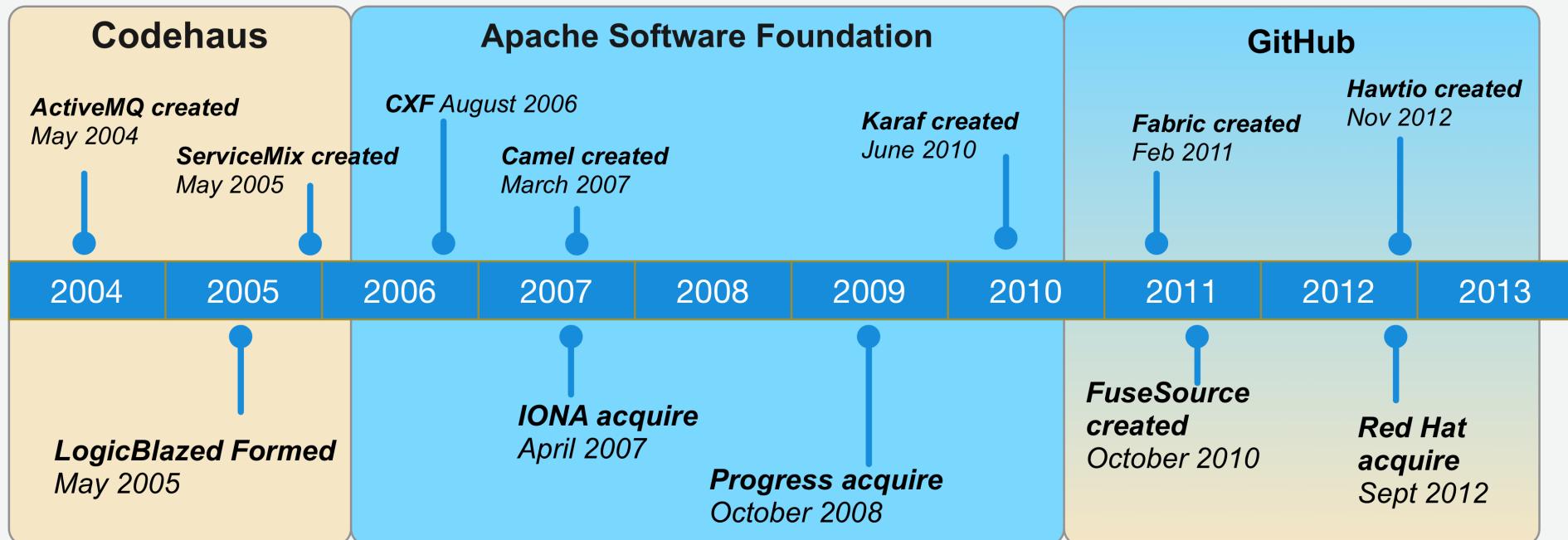
Blog: <http://cmoulliard.github.io>

Twitter: @cmoulliard

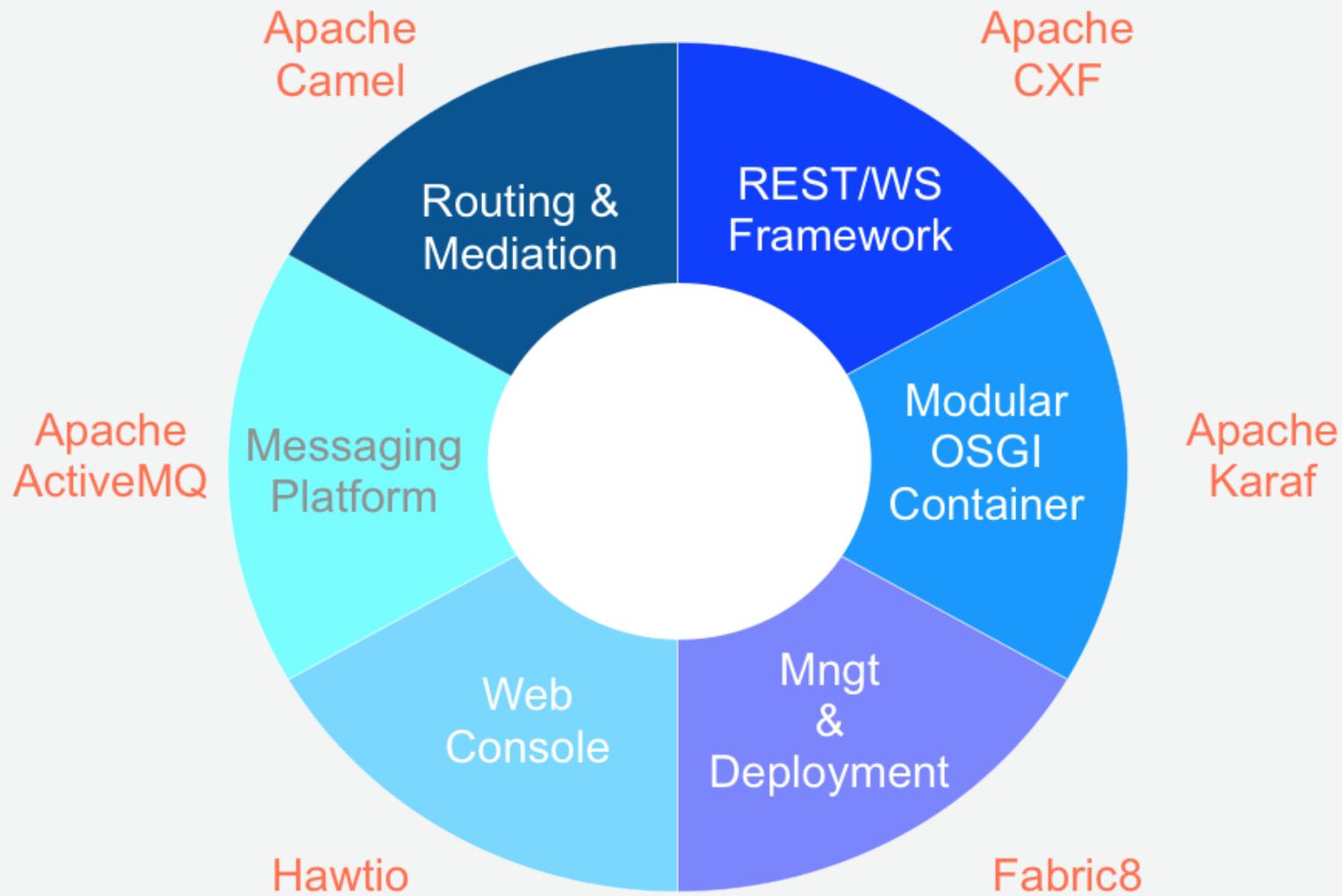
Email: cmoulliard@redhat.com

- Committer on Apache Camel, Karaf, Fabric8, Hawtio ... & PMC
- Technology evangelist
- Mountain Biker, Blogger

Long History behind Fuse

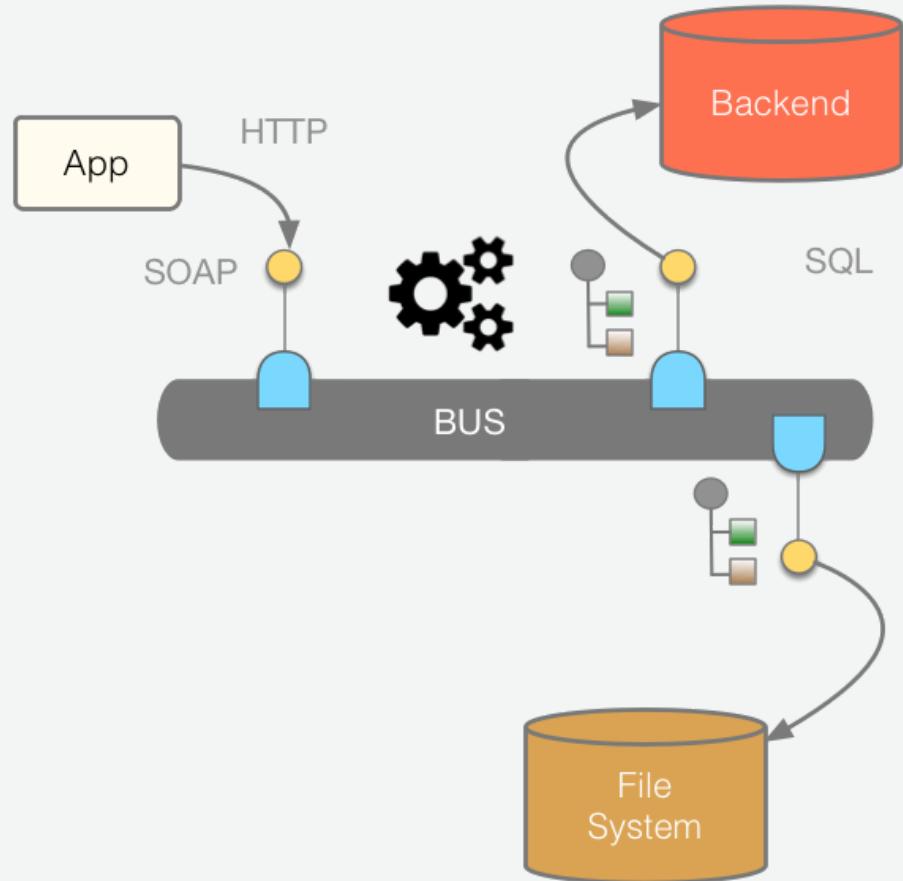


Fuse Integration Technology



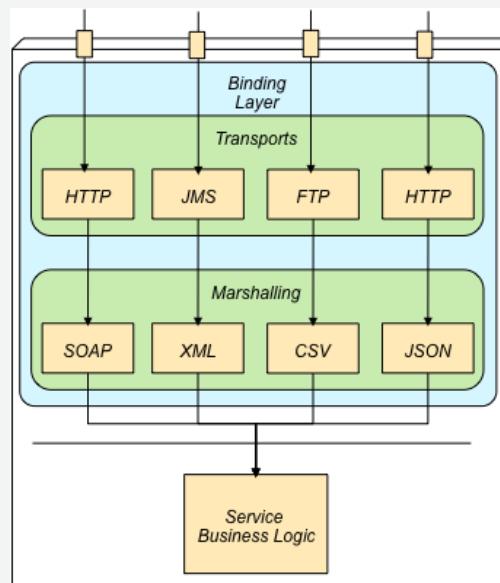
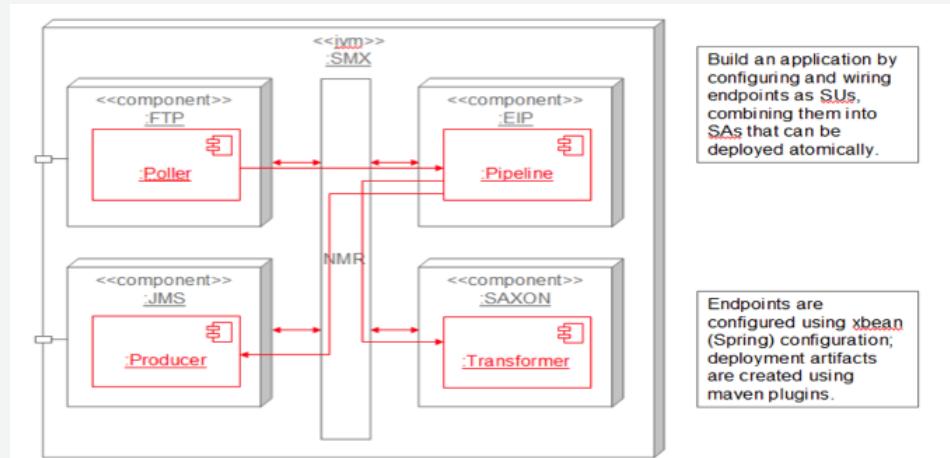
Integration platform

-  **Manage Complex use cases →**
correlation, orchestration, routing,
mediation, transformation
-  **Provide BUS →** exchange
messages
-  Designed around different specs:
JBI, SCA



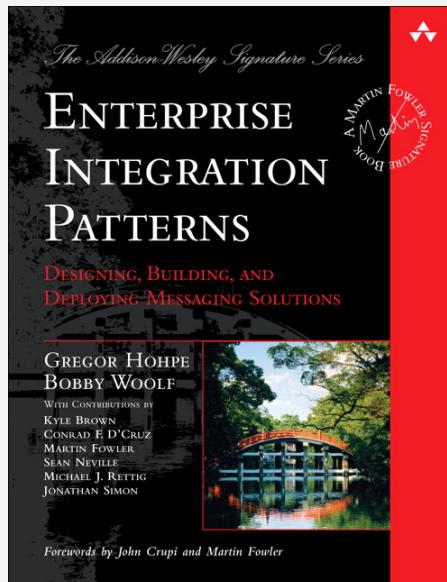
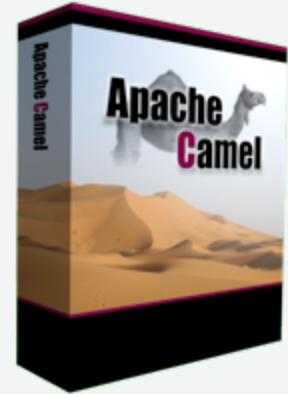
ESB & JBI

- Services & Components communicate using **NMR** bus
- Packaged as zip (=SU) in a big zip (=SA) including xml config, target service
- Constraint : messages formated as **XML**
- ! EIP not included in the spec



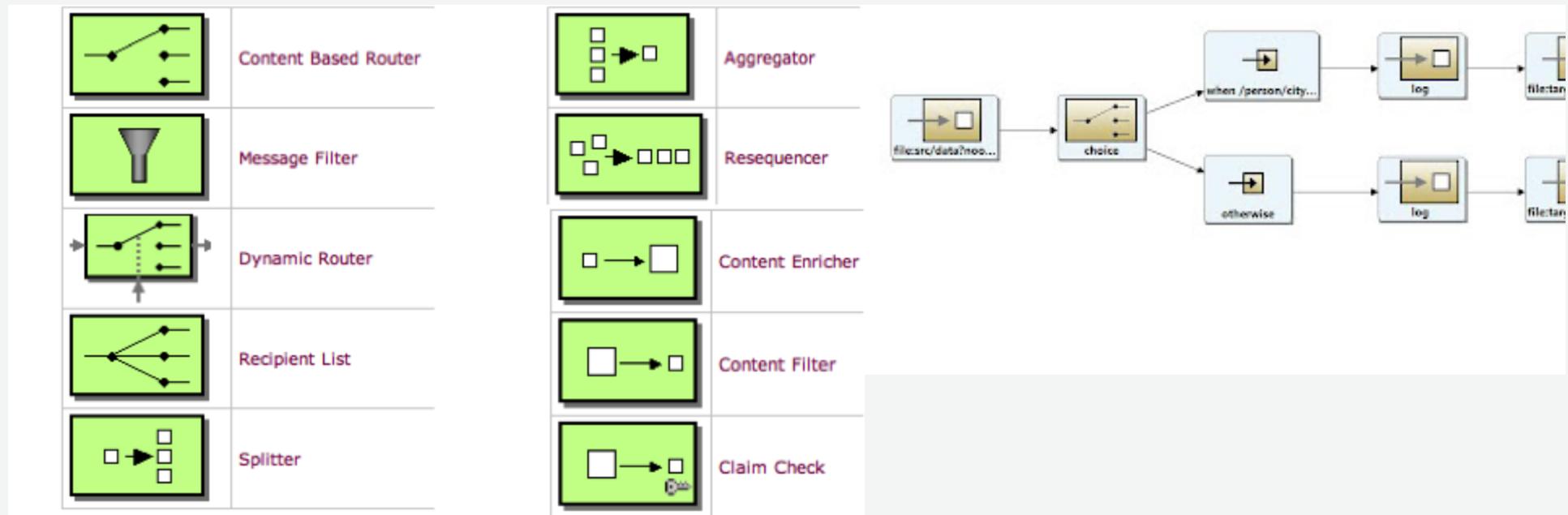
Apache Camel

- Java **Integration** Framework
- Implements → **Domain Specific Language**
- Supports **Enterprise Integration Patterns**



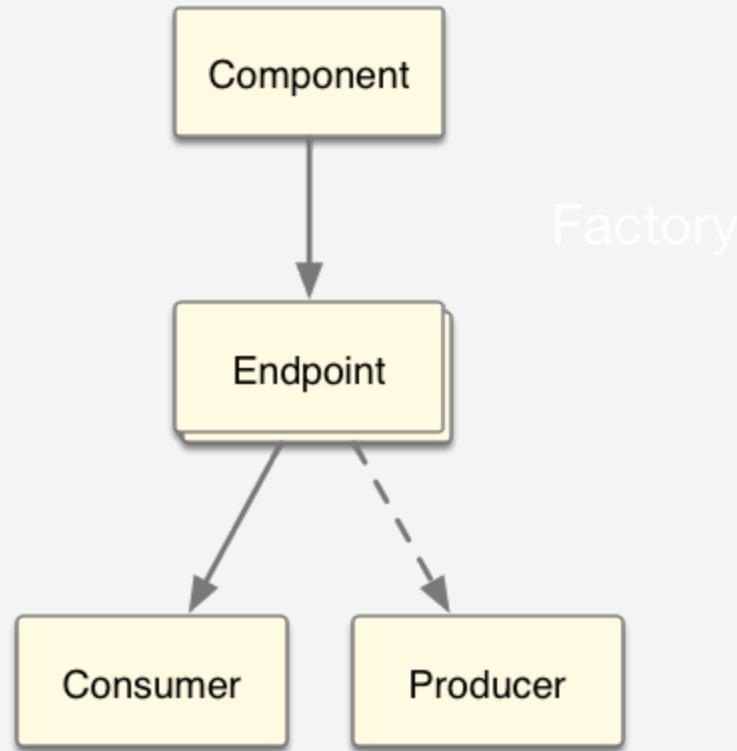
Enterprise Patterns

- **> 50 patterns** implemented
- and more : Loadbalancer, Throttler, Delayer, ...



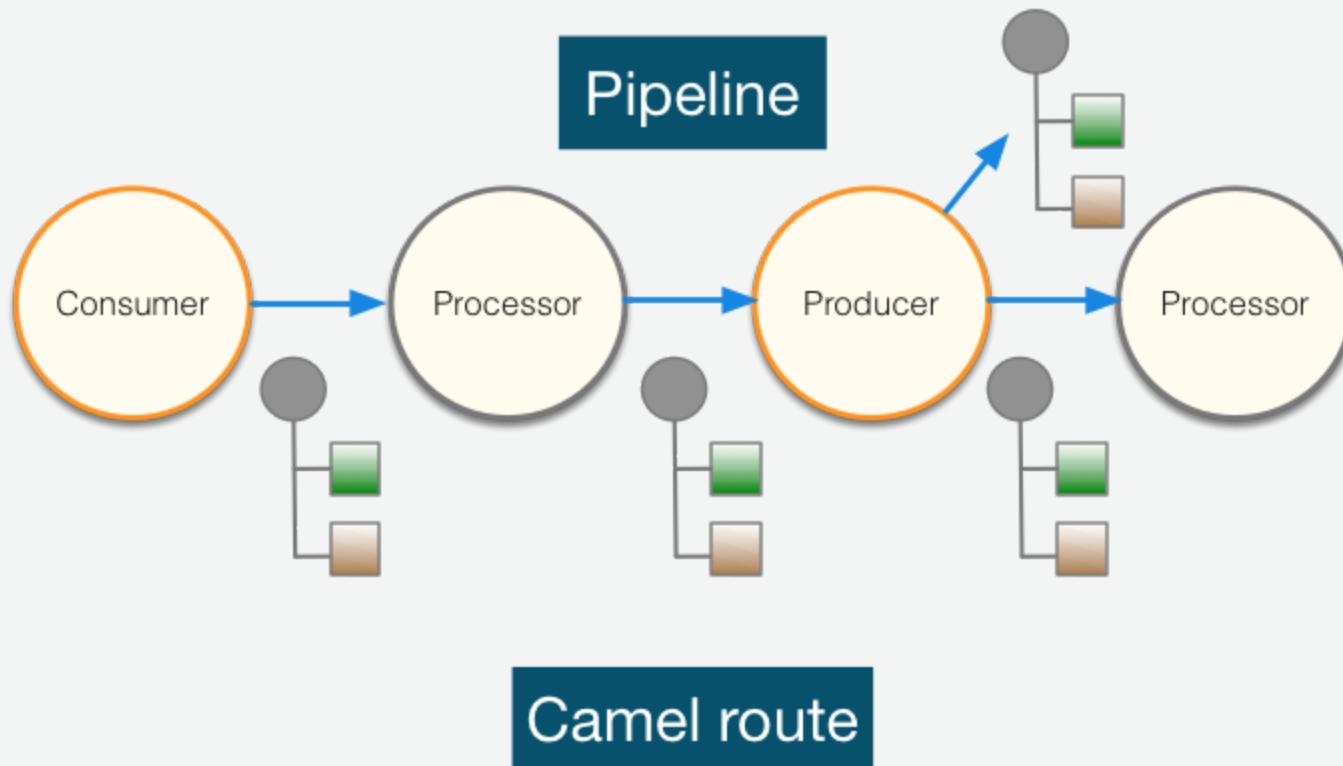
Key concepts

- Component
- Endpoint
- Consumer
- Producer



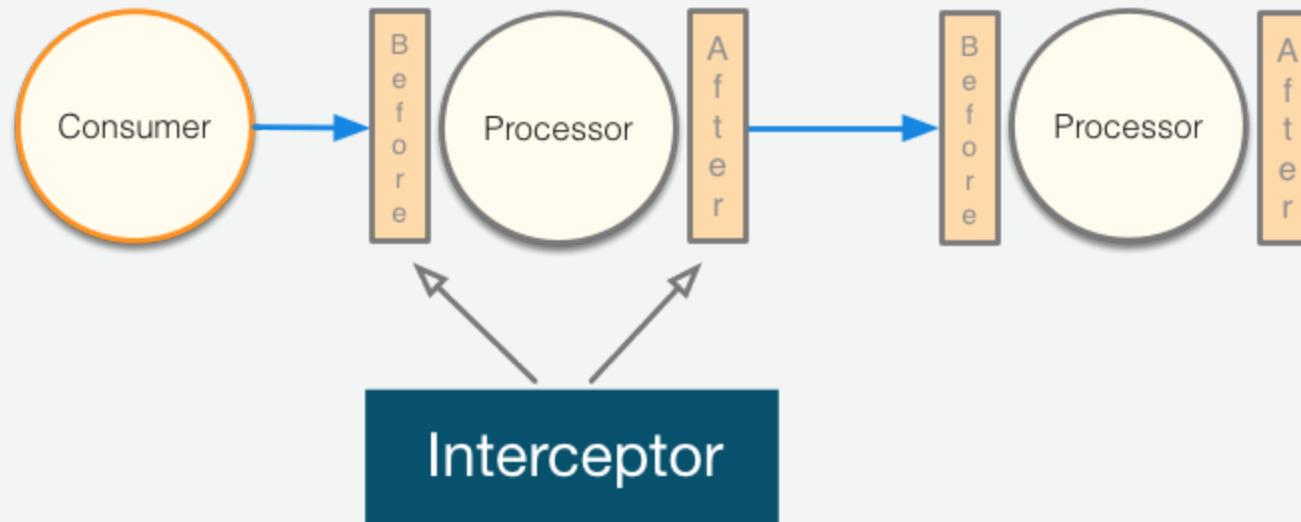
Route, processor

- Camel project ➔ Collection of routes
- Route » Processor(s) + Interceptor(s)
- Producing or consuming Messages/Exchanges



Interceptor

- To trace, log, capture business events



Convert Type

- **Type Converter** Strategy
- Allow to **convert** the body payloads from one type to another
- To and From these types
 - File
 - String
 - byte[] and ByteBuffer
 - InputStream and OutputStream
 - Reader and Writer

Type Converter Registry	
Key	Converter
(to, from)	Conv 1
(to, from)	Conv 2
.	
.	
.	

Data Format

- **Data Transformation** for complex use case

```
package my.cool.demo;

import java.io.InputStream;
import java.io.OutputStream;
import org.apache.camel.Exchange;

public interface DataFormat {

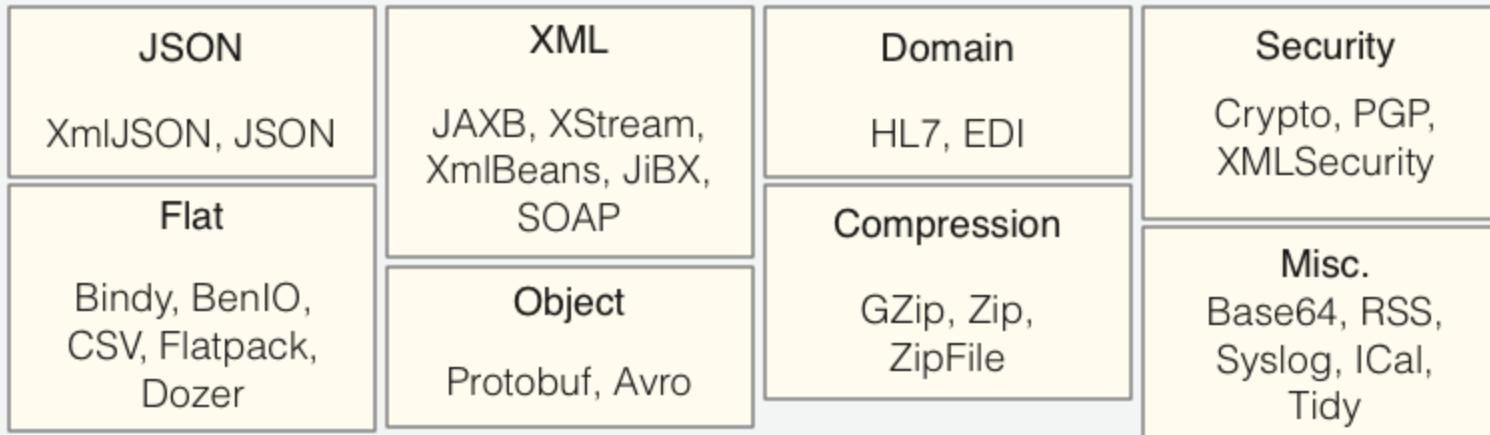
    void marshal(Exchange exchange, Object graph, OutputStream stream) throws Exception;

    Object unmarshal(Exchange exchange, InputStream stream) throws Exception;
}
```

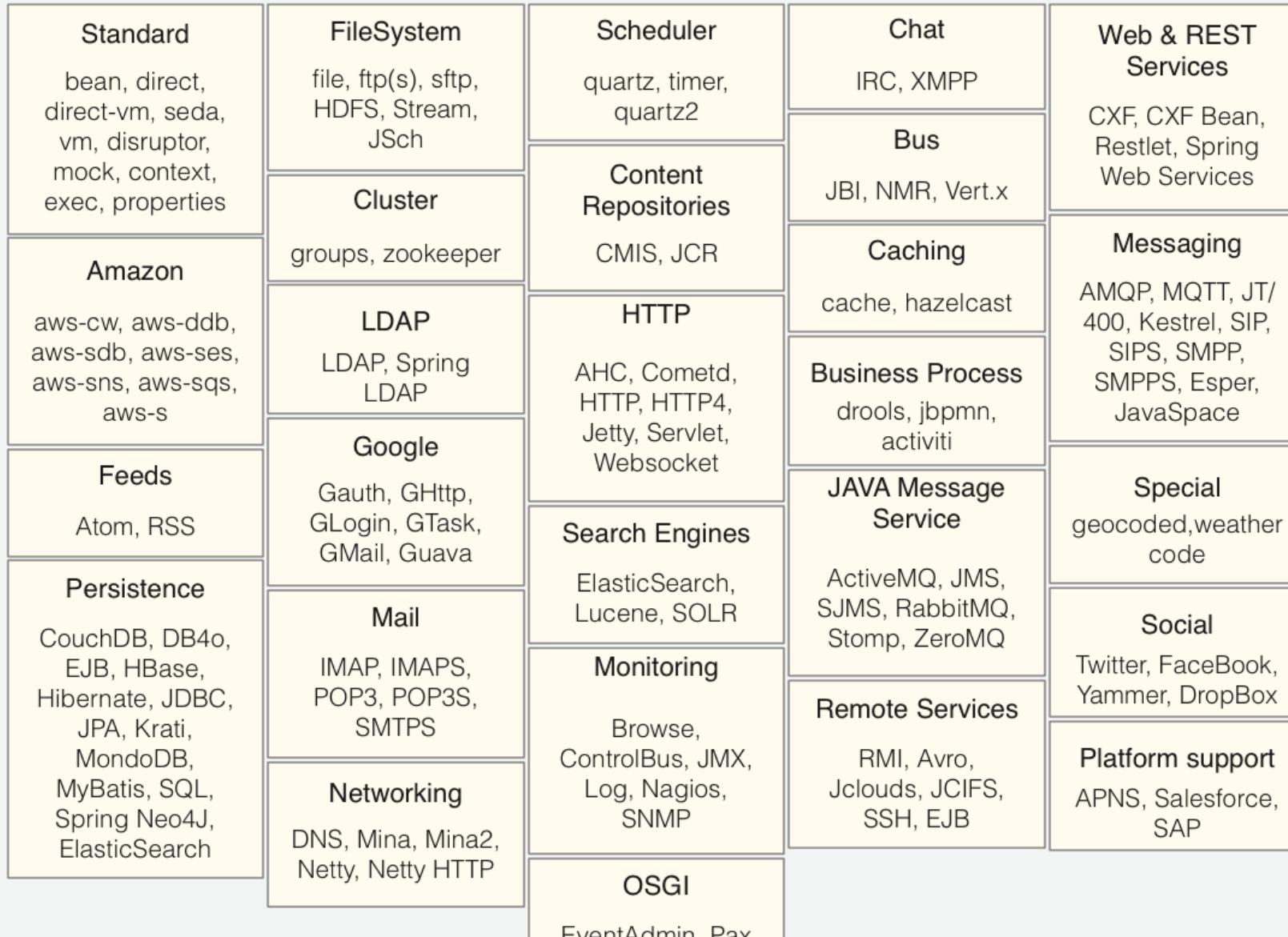
- **Marshalling** : Object → XML (JAXB)
- **Unmarshalling** : XML → Object (JAXB)

Data Format supported

- > 20 Data Format

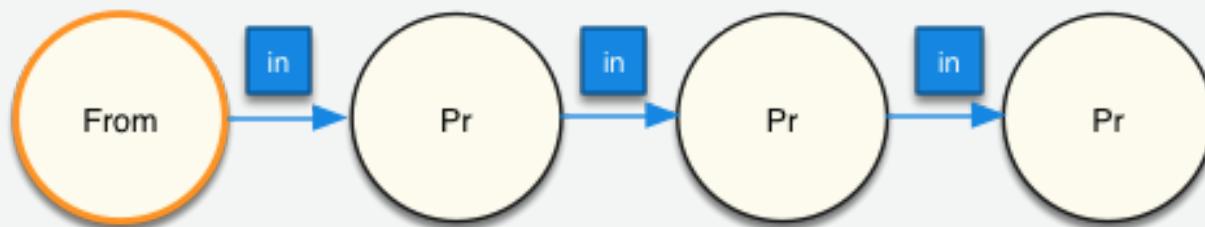


Components



Fire / Forget pattern

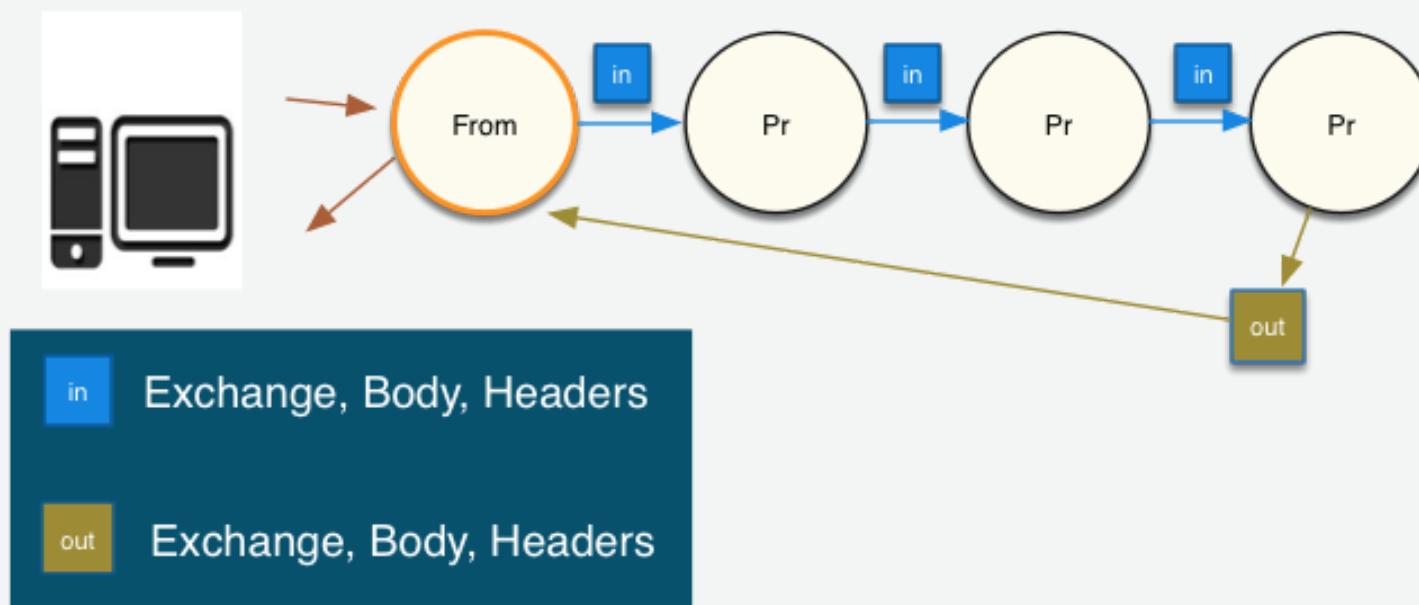
InOnly



Exchange, Body, Headers

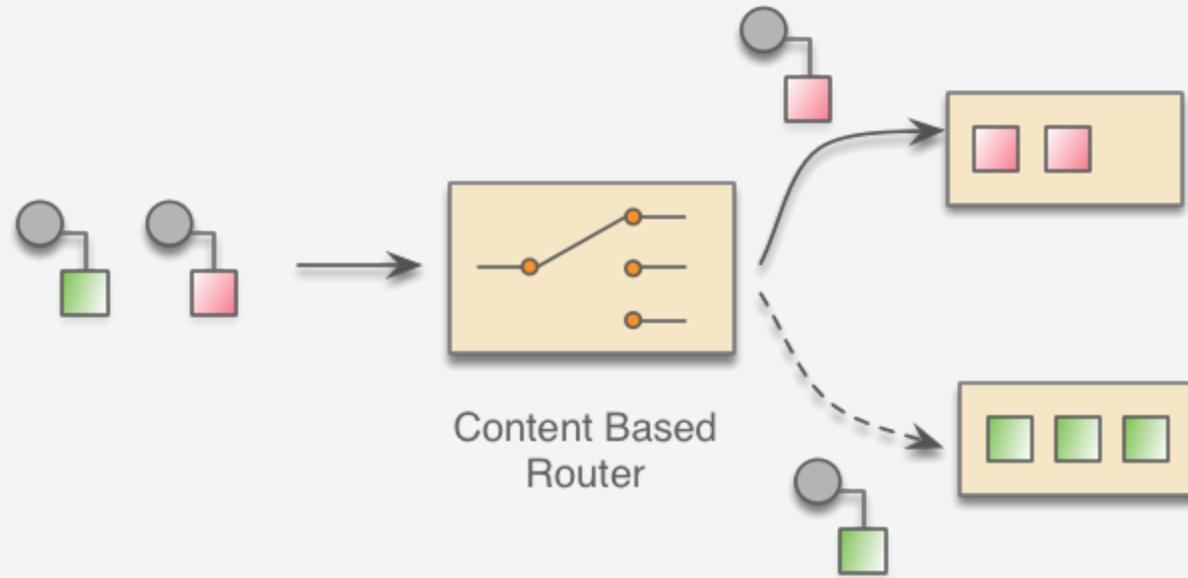
Request / Reply pattern

InOut



How To

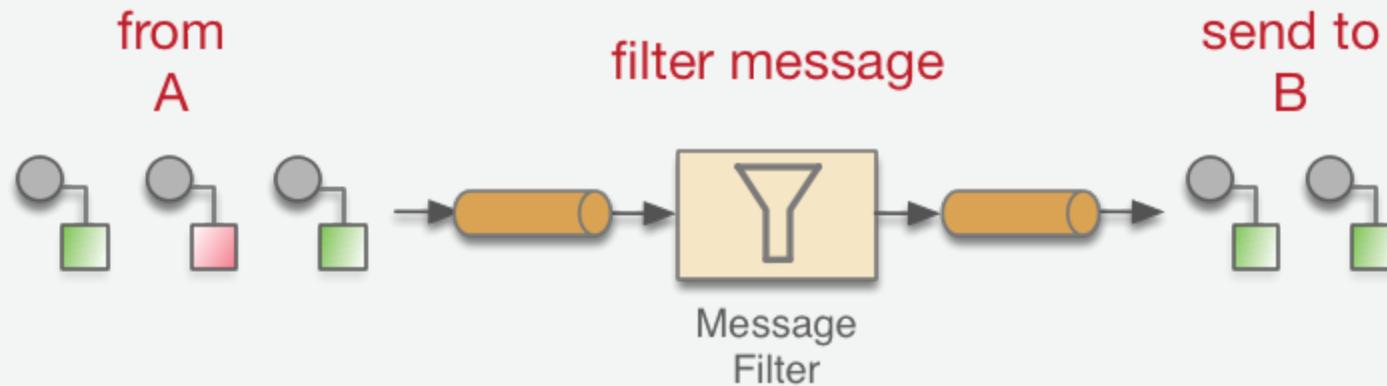
Pattern



Result

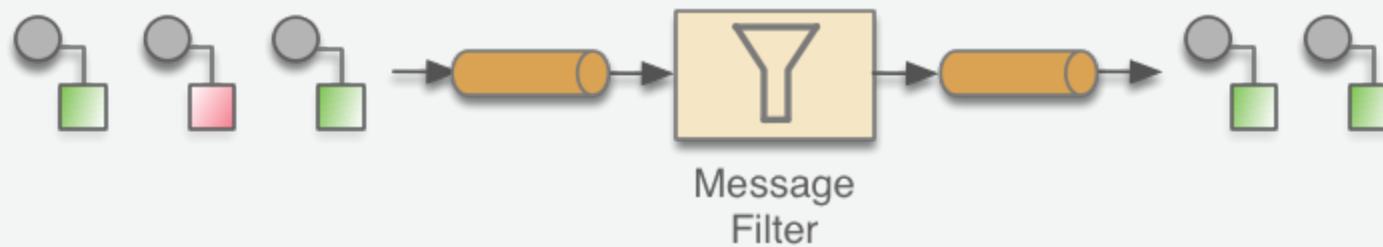
```
from("activemq:queue:quotes")
    .choice()
        .when().xpath("/quotes/product = 'widget'")
            .to("direct:b")
        .when().xpath("/quotes/product = 'gadget'")
            .to("direct:c");
```

How To



How To

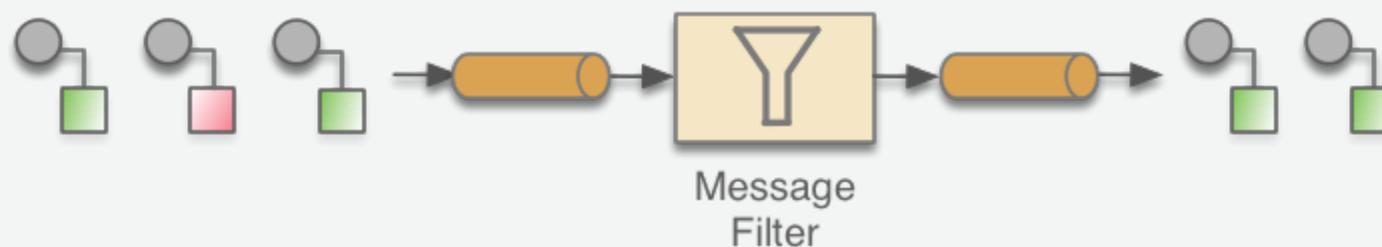
`from(A).filter(isWidget).to(B);`



How To

```
isWidget = xpath("/quotes/product = 'widget'");
```

```
from(A).filter(isWidget).to(B);
```



How To

Endpoint A = endpoint("activemq:queue:all");

Endpoint B = endpoint("activemq:queue:widget");

Predicate isWidget = xpath("/quotes/product = 'widget'");

from(A).filter(isWidget).to(B);

How To

```
public void configure() throws Exception {
    Endpoint A = endpoint("activemq:queue:all");
    Endpoint B = endpoint("activemq:widget");
    Predicate isWidget = xpath("/quote/product = 'widget'");
    from(A).filter(isWidget).toB();
}
```

Java DSL

- Fluent API

```
import org.apache.camel.builder.RouterBuilder;

public class FilterRoute extends RouteBuilder {
    public void configure() throws Exception {
        Endpoint A = endpoint("activemq:queue:all");
        Endpoint B = endpoint("activemq:widget");
        Predicate isWidget = xpath("/quote/product = 'widget'");

        from(A).filter(isWidget).to(B);
    }
}
```

XML DSL

- Spring, Blueprint

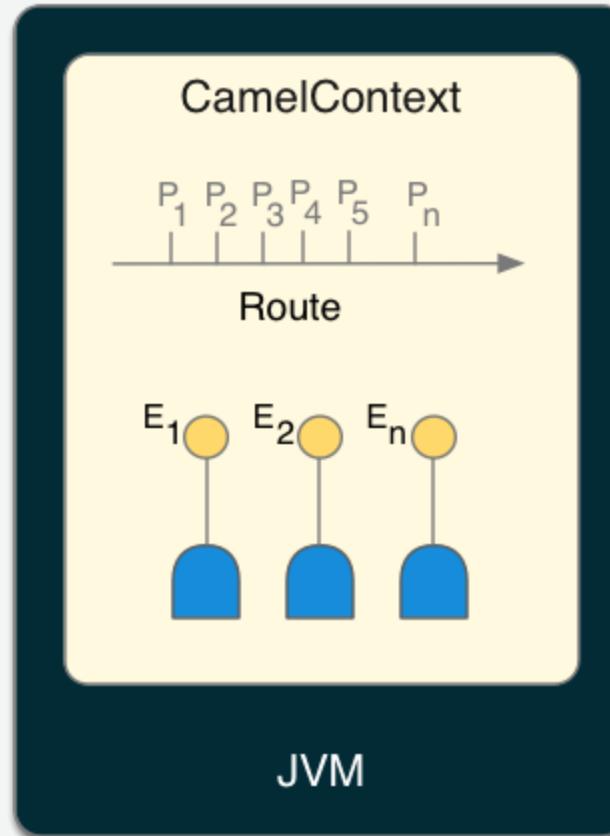
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
           http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

    <bean id="quotesService" class="my.cool.demo.camel.QuotesService"/>

    <camelContext xmlns="http://camel.apache.org/schema/spring">
        <route>
            <from uri="activemq:queue:all"/>
            <filter>
                <xpath>/quote/product/ = 'widget'</xpath>
                <bean id="quotesService" method="widget"/>
            </filter>
        </route>
    </camelContext>
```

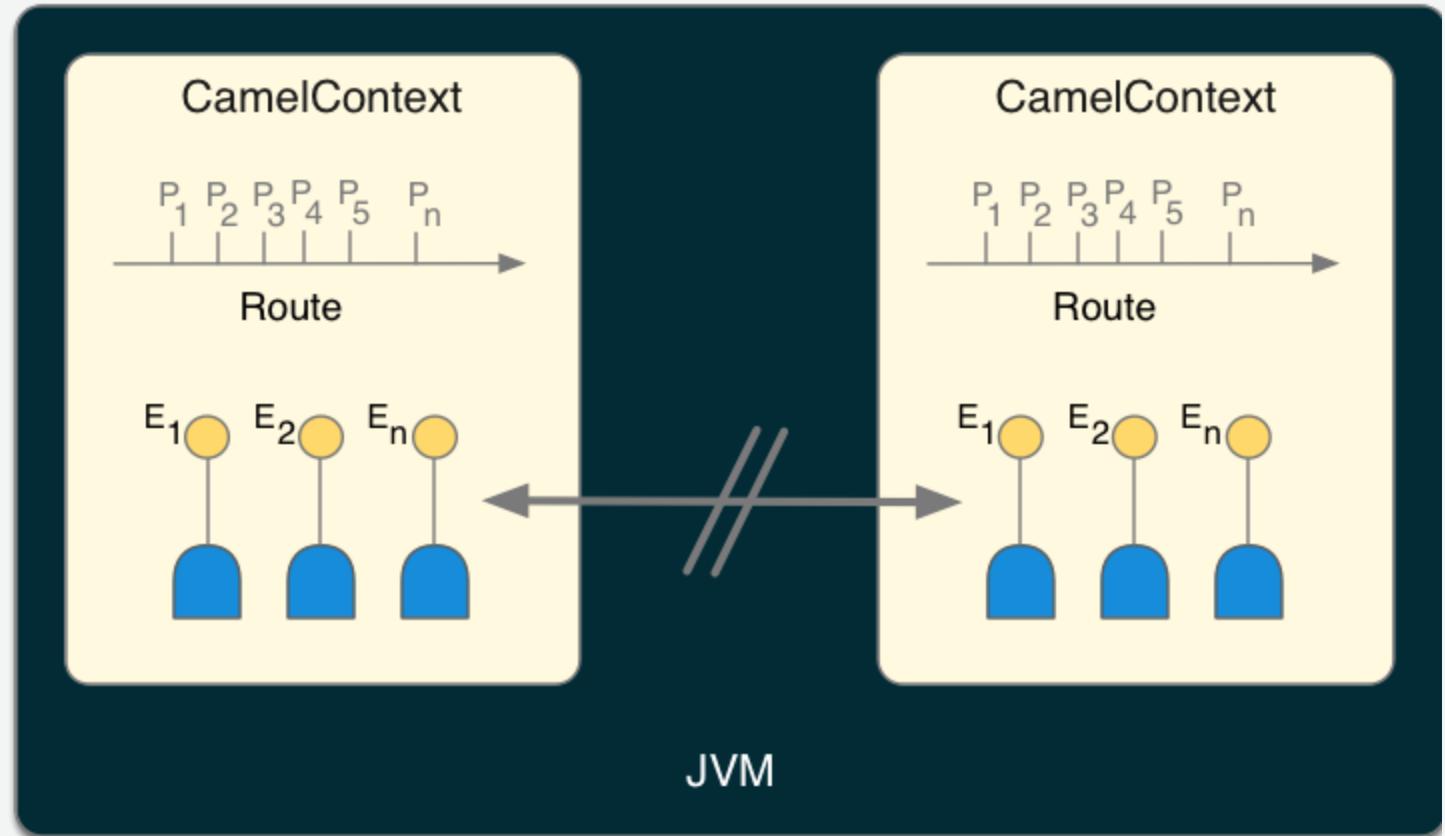
Container

- Routes/**Endpoints**
registered ➔ CamelContext
- Policy
 - Security
 - Lifecycle
- Tracing
- JMX
- Threads can be configured

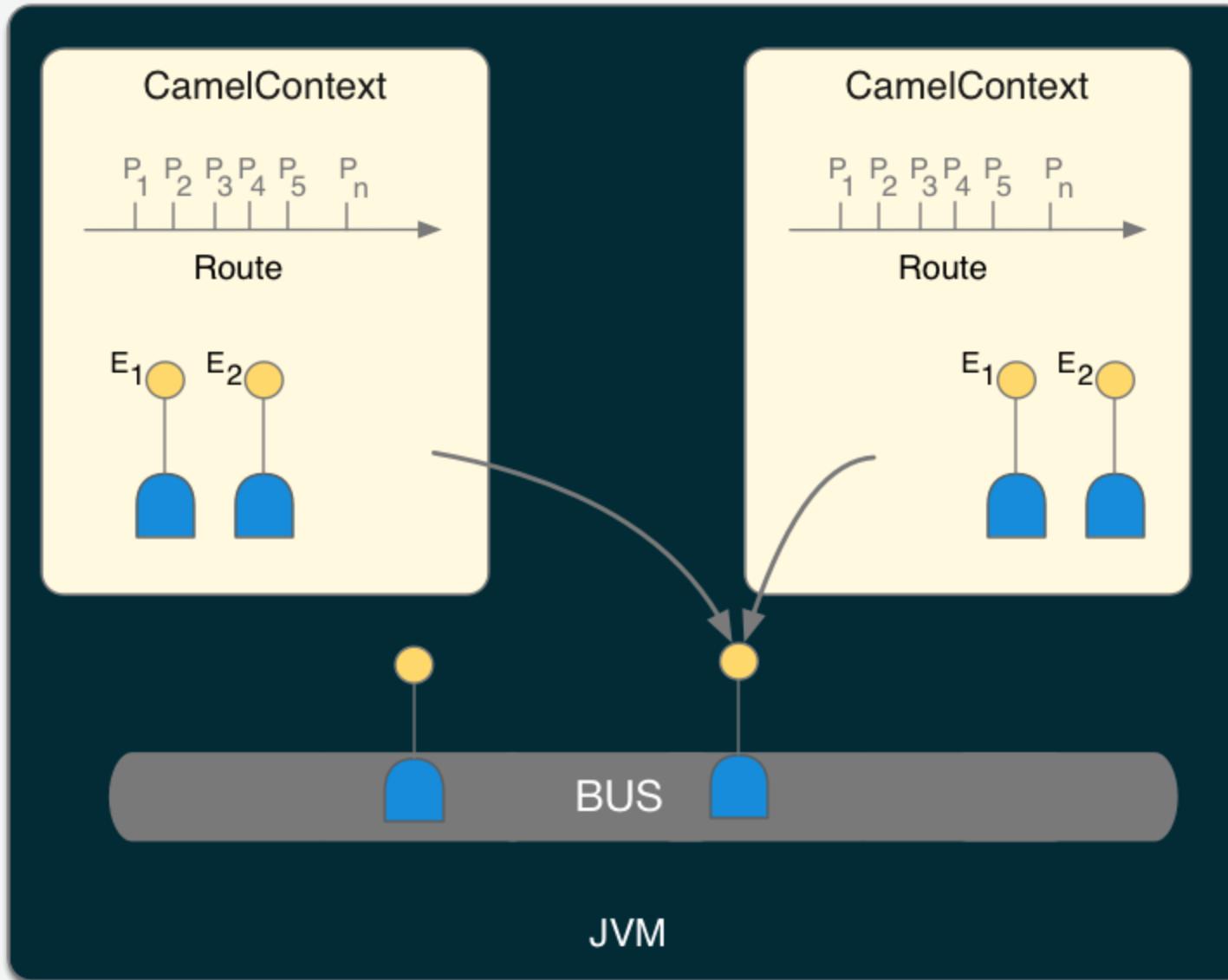


Communication

- Cross communication not allowed between context



Communication



Facts

- **In-Memory** bus
- Support **Object** : XML, File, Stream, Bytes
- **Predicate** & **Expression** language (xslt, xpath, ...)
- **Sync/Async** exchanges
- Threads Management
- **Tx** Architecture
- **Error** & **Exception** handling
- Policy driven
- Container **Agnostic**

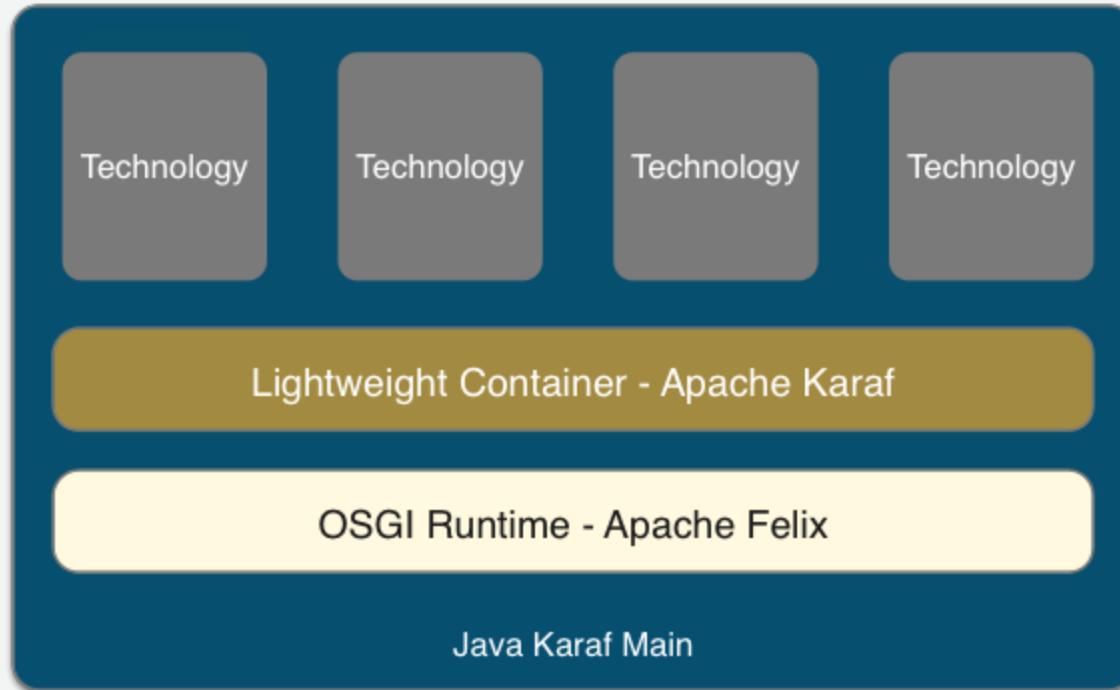
Apache Karaf

- Java **OSGI** Runtime
- Offer **modularity** for **Integration**
- **Multi-Technology** platform



meretmarine.com

Architecture



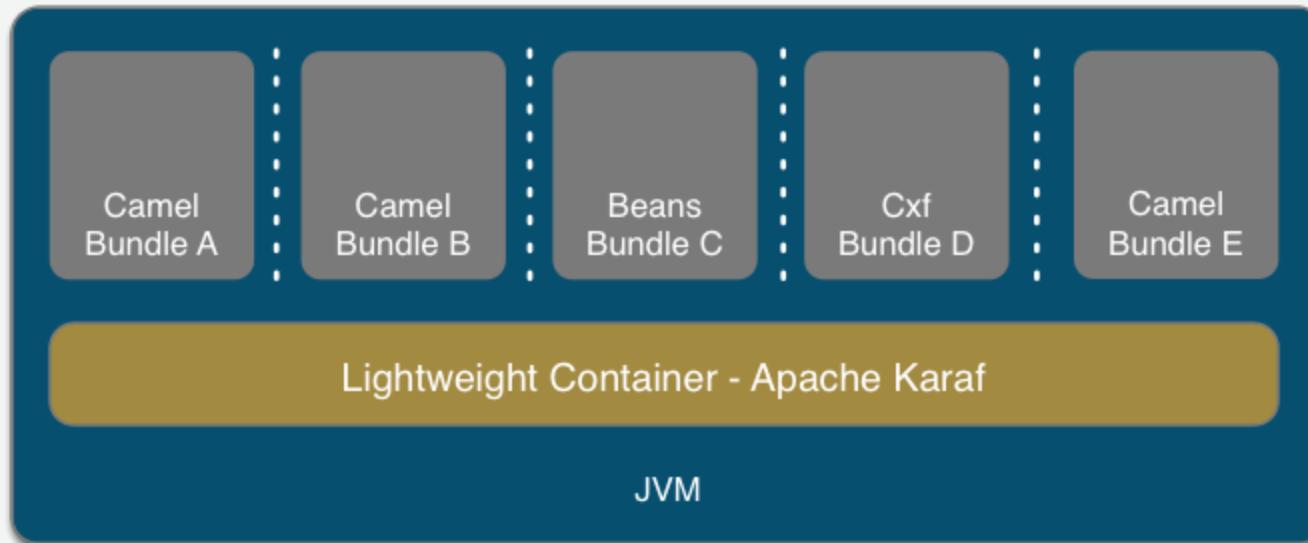
- Technology → Camel, CXF, ActiveMQ, Spring, Fabric, ...
- Modular platform → deploy or remove container/libraries

Core features

- **SSH** server
- Allow to **administrate/create** instances
- Provide **provisioning** solution features
- **Hot** deployment
- Configure & manage instances
- **JAAS** Security layer
- Role Base Access Control (RBAC)

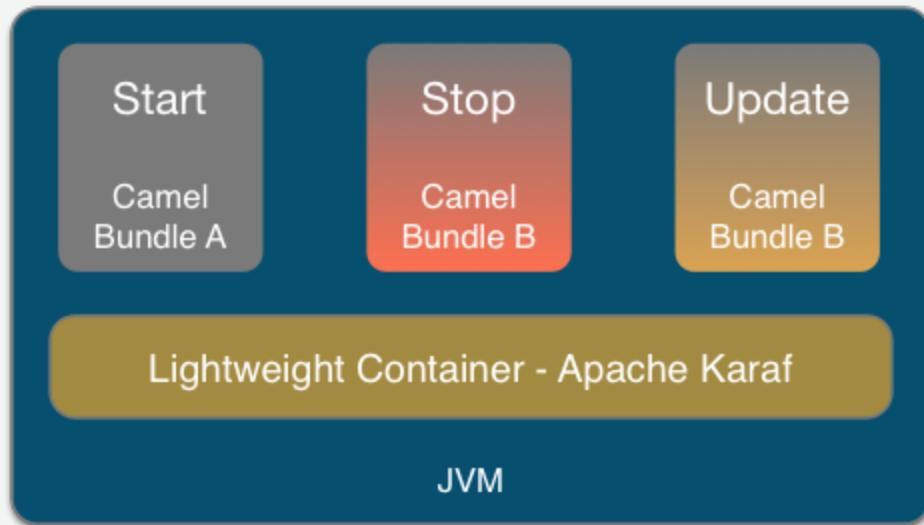
Integration container

- Camel routes isolated from each other (classloader)
- Bundle → CamelContext boundary → acting as a Local BUS
- Camel routes → can have different SLA (Threads, Policies, ...)



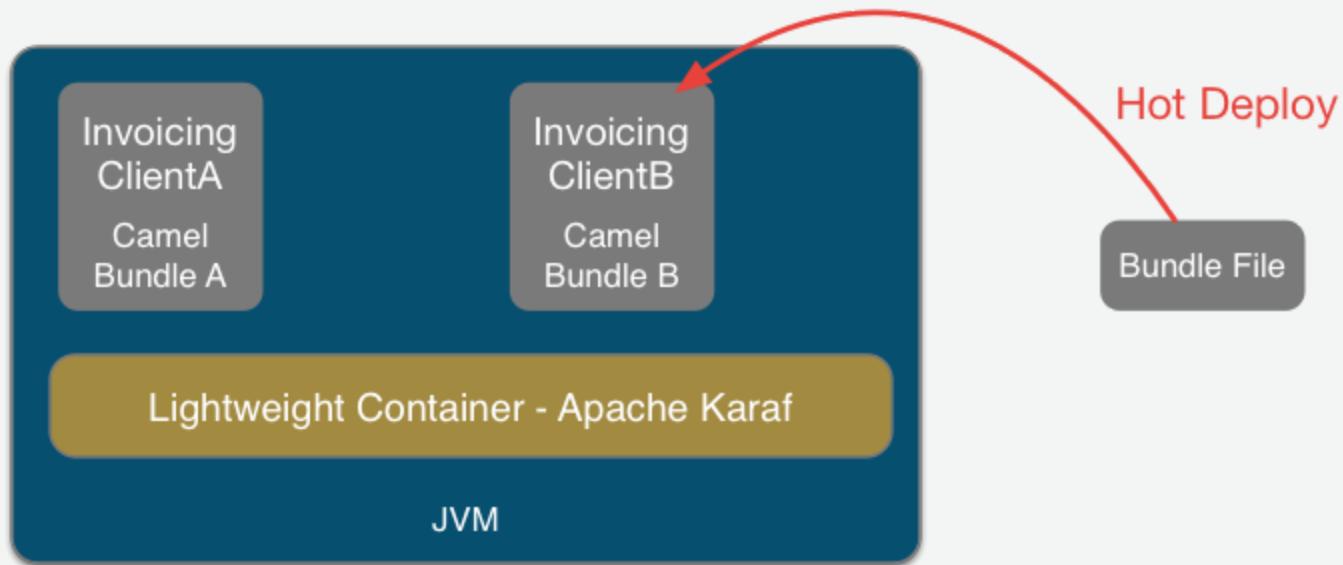
Integration container

- Camel routes can be started/stopped/updated
- → Simplify maintenance process

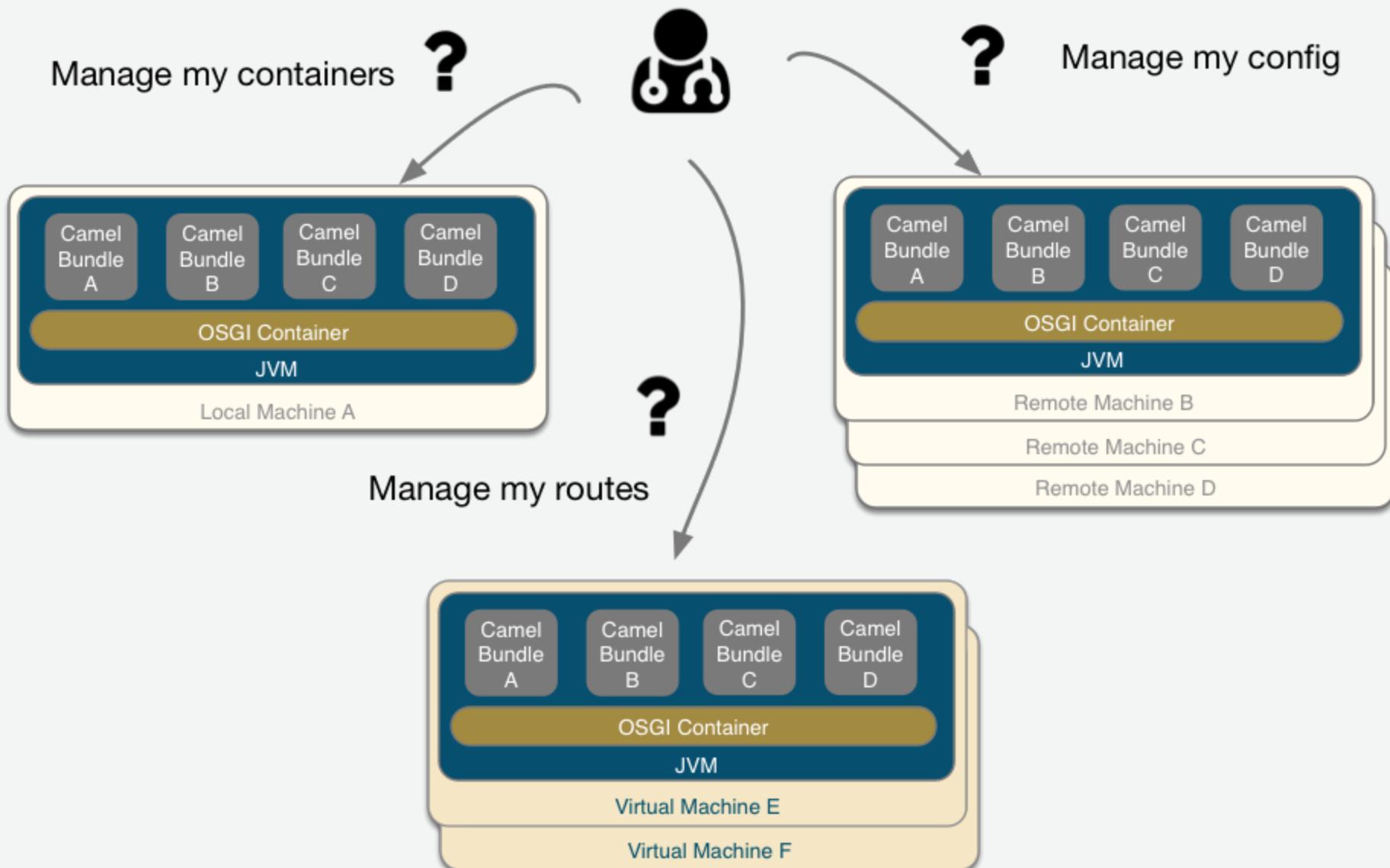


Integration container

- New routes can be **hot deployed**
- Like also "Beans/POJO, Web Services, ..."

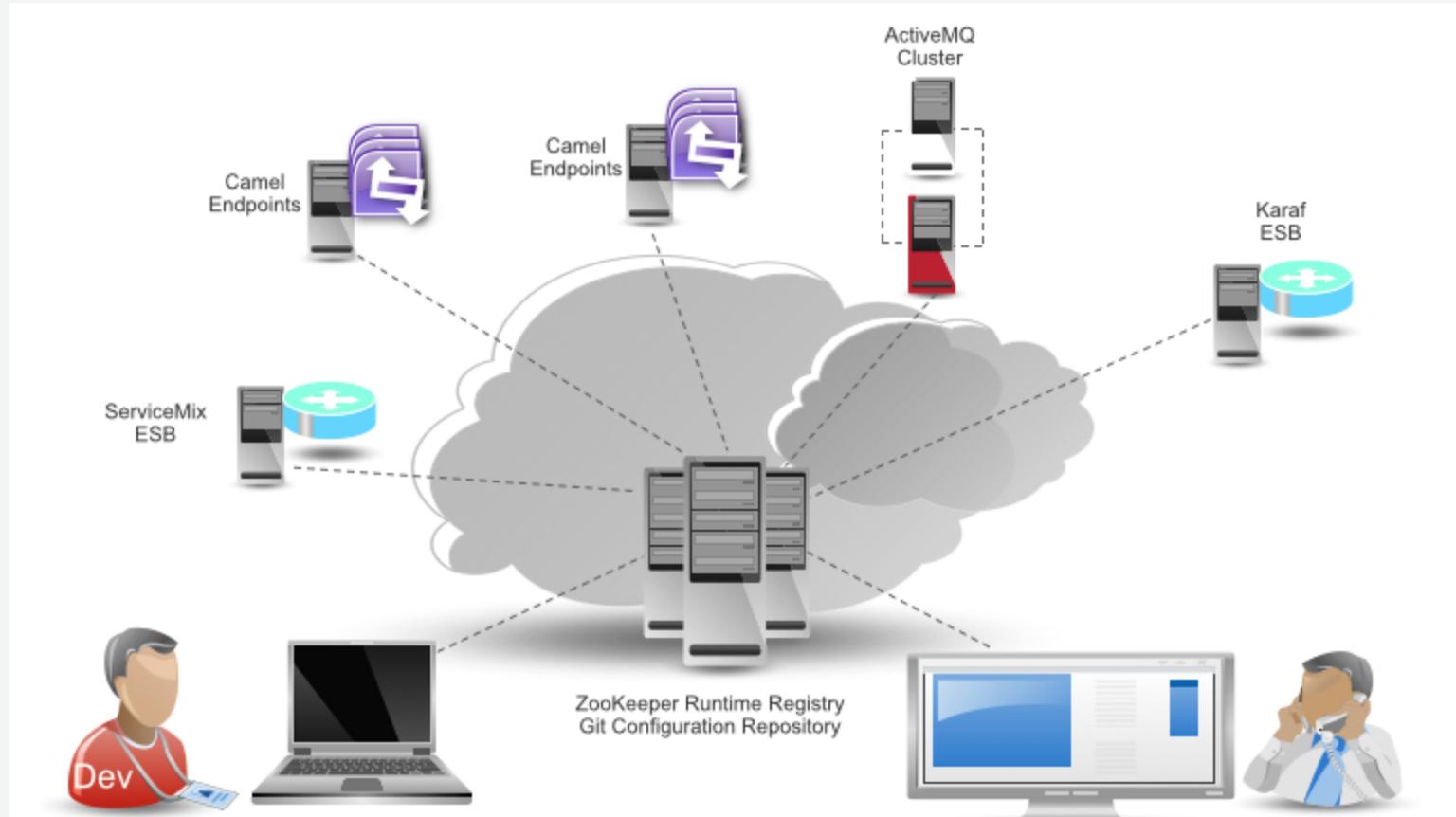


(Cloud) Deployment



Fabric8 v1

- Opensource integration project - <http://fabric8.io>
- Mission → simplify management & deployment java **integration** services on **different** machines & JVMs

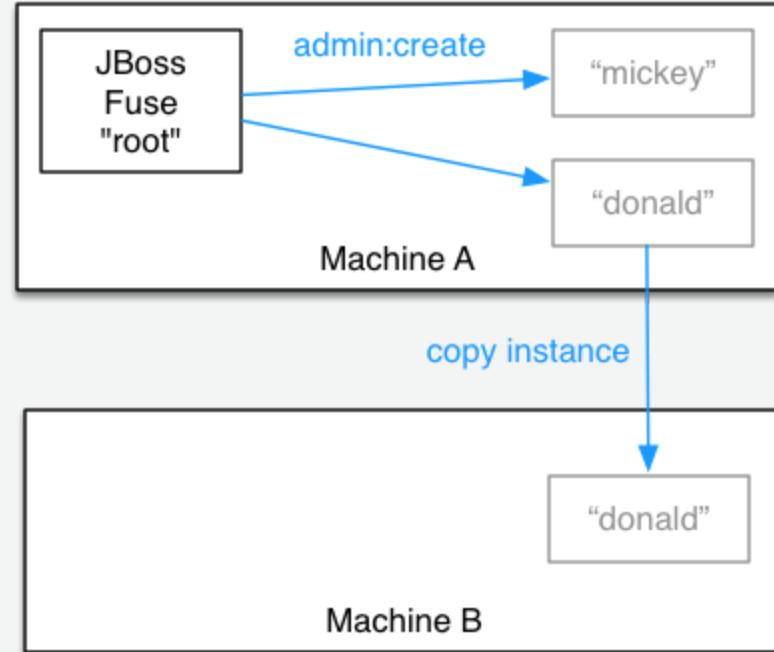


Features

- **Manage** container creation (locally, remotely, cloud, openshift, docker, ...)
- **Visualise** what is running into JVM to understand your platform
- **Monitor** whats running and easily scaling up or down
- Support **Upgrade** via **Version changes** and Rollback
- **Virtualize** services (endpoints), processes
- **Search** and **storage** engine for logs, camel, messages, metrics

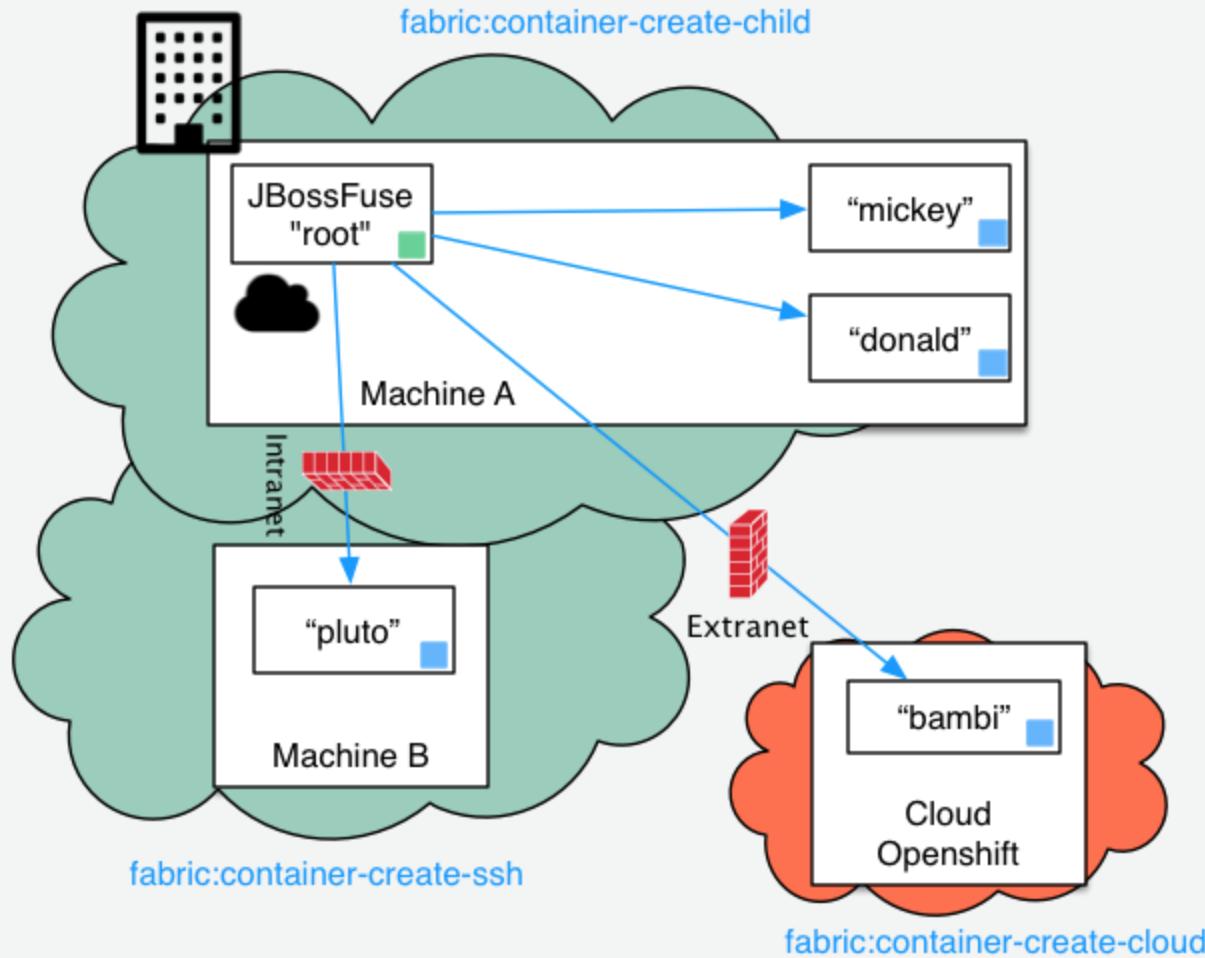
Karaf limitations

- Karaf can create new instances (locally) & administrate them (locally or remotely)



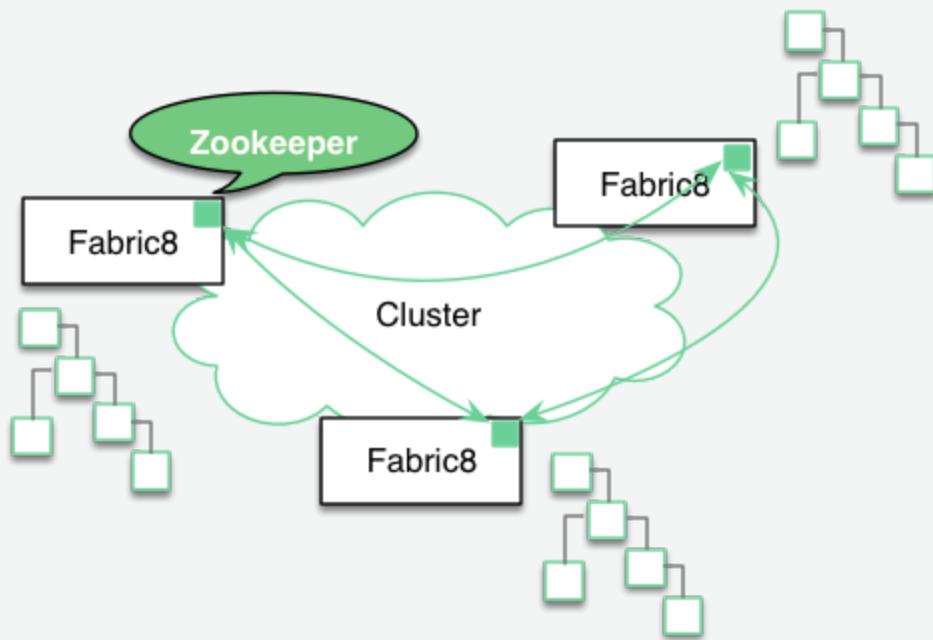
- Instances are **not cloned**
- **Configurations** must be managed (manually, script)

Fabric extends the possibilities



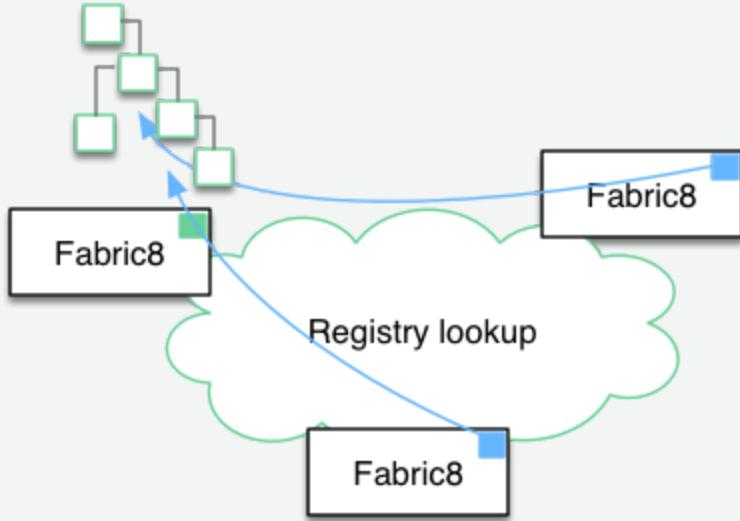
Coordinating System : Zoo

- **Rely on ➔** Zookeeper server (ensemble of 1,3, 5 or servers) 
- **Coordinating distributed** systems in a **reliable** way (electing leaders, implementing master/slave, sharding or federation of services).



Fabric Agents

- Are the **clients** of the Zookeeper server(s)



- They will communicate with Zk server to :
- → register container info (ports, services, endpoints, processes)
- → get their provisioning

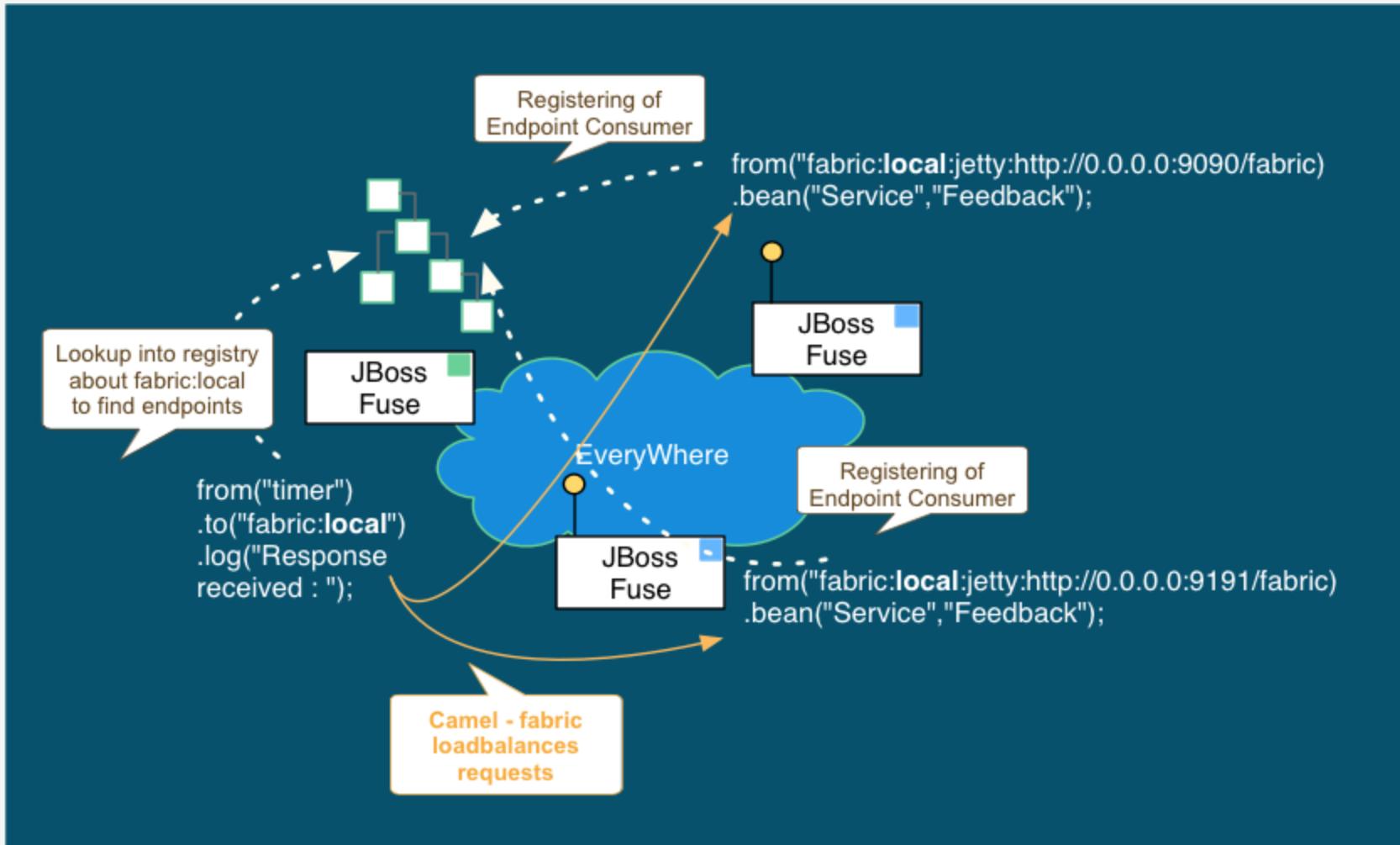
Profiles

- **Behavior** of a container
- **Envelope(s)** containing **artifacts** to be deployed, **parameters** (system, jvm, services) to be configured
- Can be **versioned**, facilitate mngrt - rollback

The screenshot shows the Red Hat JBoss Fuse Management Console interface. The URL in the browser is `localhost:8181/hawtio/index.html#/wiki/branch/1.0/view/fabric/profiles/feature/cxf.profile`. The top navigation bar includes links for Runtime, Wiki, Dashboard, and Health, along with a user dropdown for 'admin'. The main content area displays the 'feature-cxf' profile configuration. On the left, there's a sidebar with options like 'Up a directory' and 'io.fabric8.agent.properties'. The main panel has tabs for 'Features (3)', 'Feature Repositories (1)', 'Bundles (0)', and 'FABs (0)'. The 'Features' tab is selected, showing a list with 'fabric-cxf-registry' at the top, followed by 'cxf' and 'war'. Below this, under 'Parents:', 'karaf' is listed with a 'Remove' icon. Under 'Children:', several quickstarts are listed: 'example-quickstarts-soap', 'example-quickstarts-secure.rest', 'example-quickstarts-rest', 'example-quickstarts-secure.soap', and 'example-cxf'. There are also 'Refresh', 'Copy', 'Assign', and 'New' buttons at the top right of the main panel.

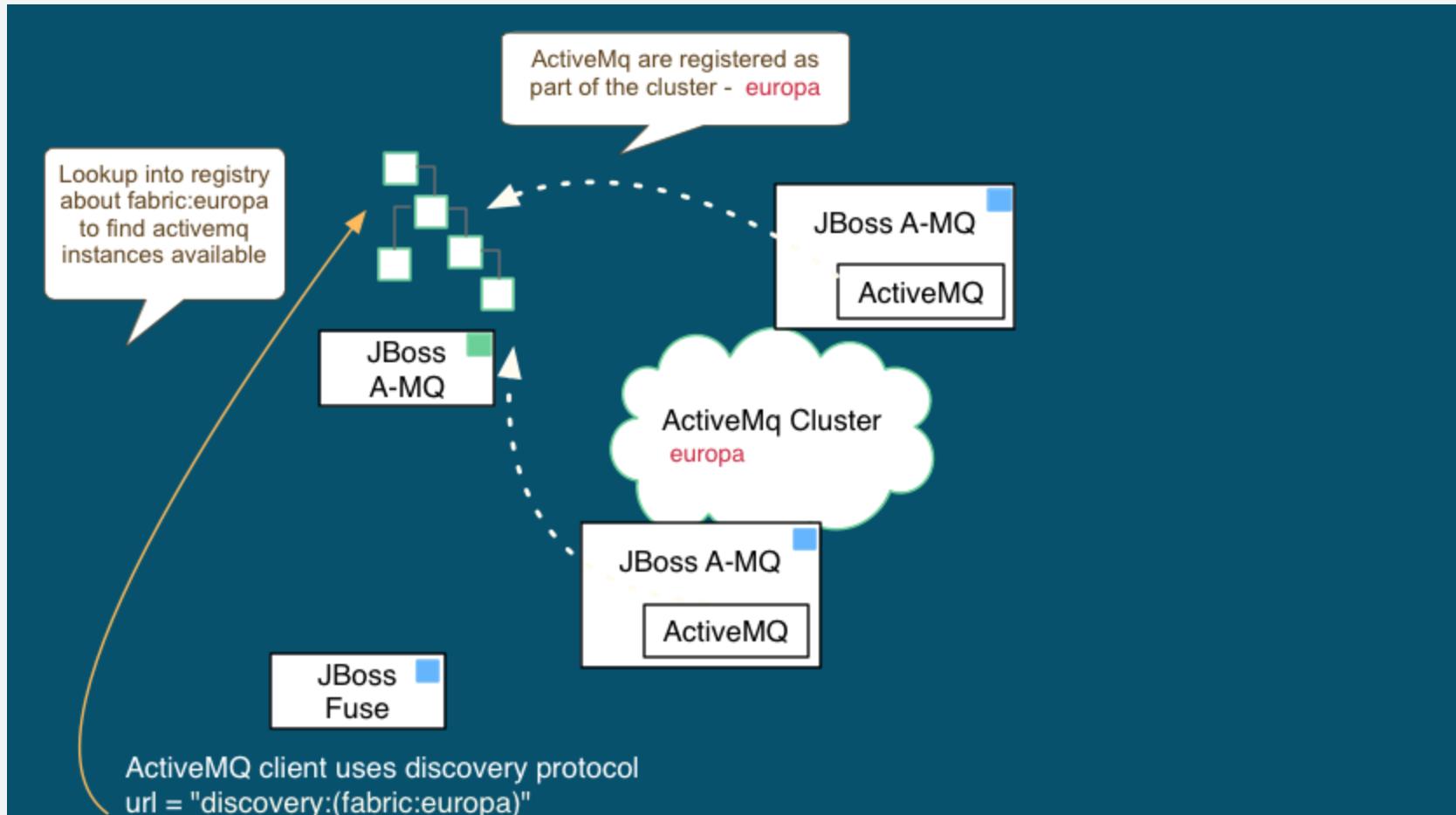
Virtualization & Load balancing

- Goal ➔ Create **virtual** endpoints, to **scale** services (WS/REST, A-MQ)



New topologies

- New **topologies** (Replicated - **LevelDB** storage, **NPlus1**),
- Broker **discovery**



Metrics

- Fabric Insight Technology →
 - NoSQL storage for JSON documents
 - Based on ElasticSearch



Dashboard

- **Kibana** is the web front end



- **Full Text Search** features →



Info collected

- Logs, **Camel** metrics, **JMX** metrics, your own **data/JSON** metrics

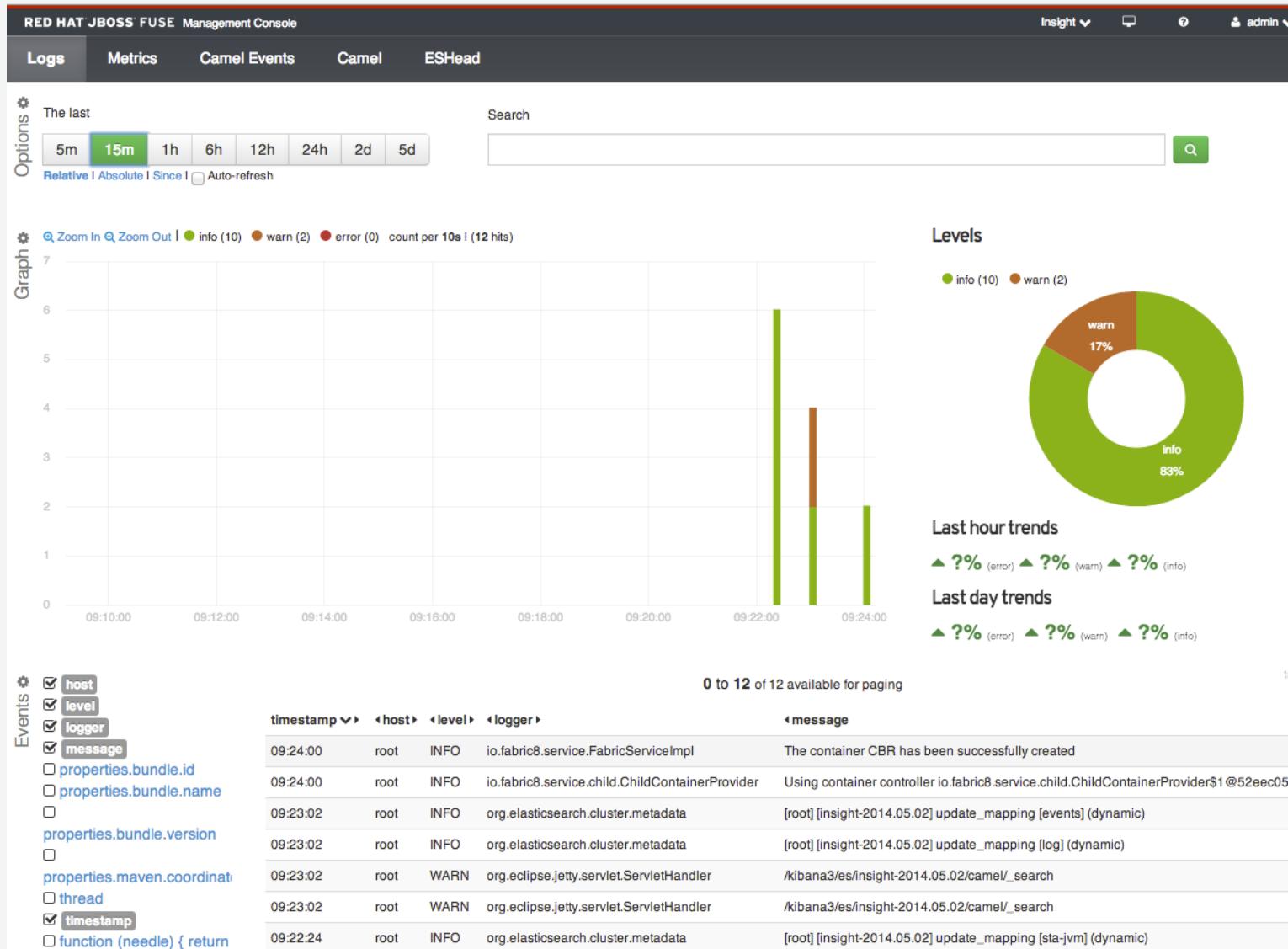
```
import org.apache.camel.Header;
import org.fusesource.insight.storage.StorageService;
import java.sql.Timestamp;
import java.util.Date;

public class StoreService {

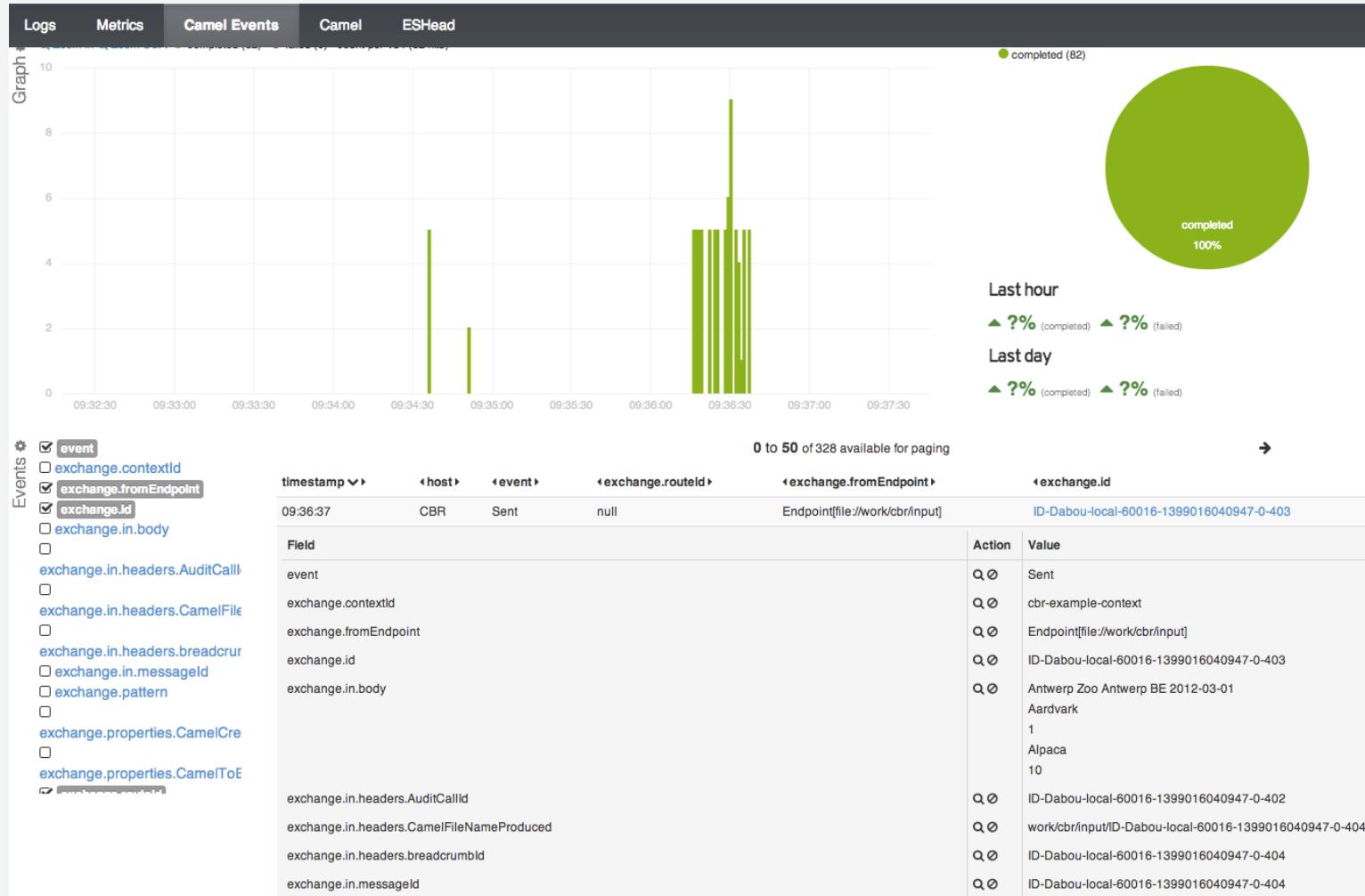
    private static String ES_TYPE = "insight-tweet";
    private static StorageService storageService;

    public static void store(@Header("tweet-full") String data) {
        storageService.store(ES_TYPE, generateTimeStamp(), data);
    }
}
```

Analyzed using kibana



Insight Camel



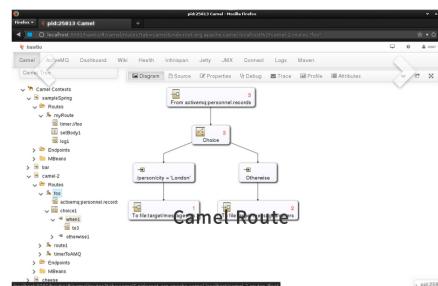


- **OpenSource** project - Apache License
- **Pluggable & modular web console** for managing Java **MBeans**

A screenshot of the Hawt.io web interface. At the top, there's a navigation bar with links for "Get Started Now!", "Documentation", "Community", "Demos", and "github". The main content area features a large heading "Qhawtio" with a flame icon, followed by the text "a modular web console for managing your Java stuff". Below this is a "View Demos" button and a "Get Started Now!" button. On the right side of the page, a window titled "pizzahut Camel" displays a Camel route diagram. The diagram shows a "From activemq:personal-records" source connected to a "Camel Route" node, which then branches into two paths: one leading to a "Dove" node and another to an "Otherwise" node. The "Camel Route" node is also connected to a "To the register" sink. The left sidebar of the window lists various Camel components like "Camel Context", "Camel Components", "Camel Beans", "Camel Test", "Camel2", and "Routes".

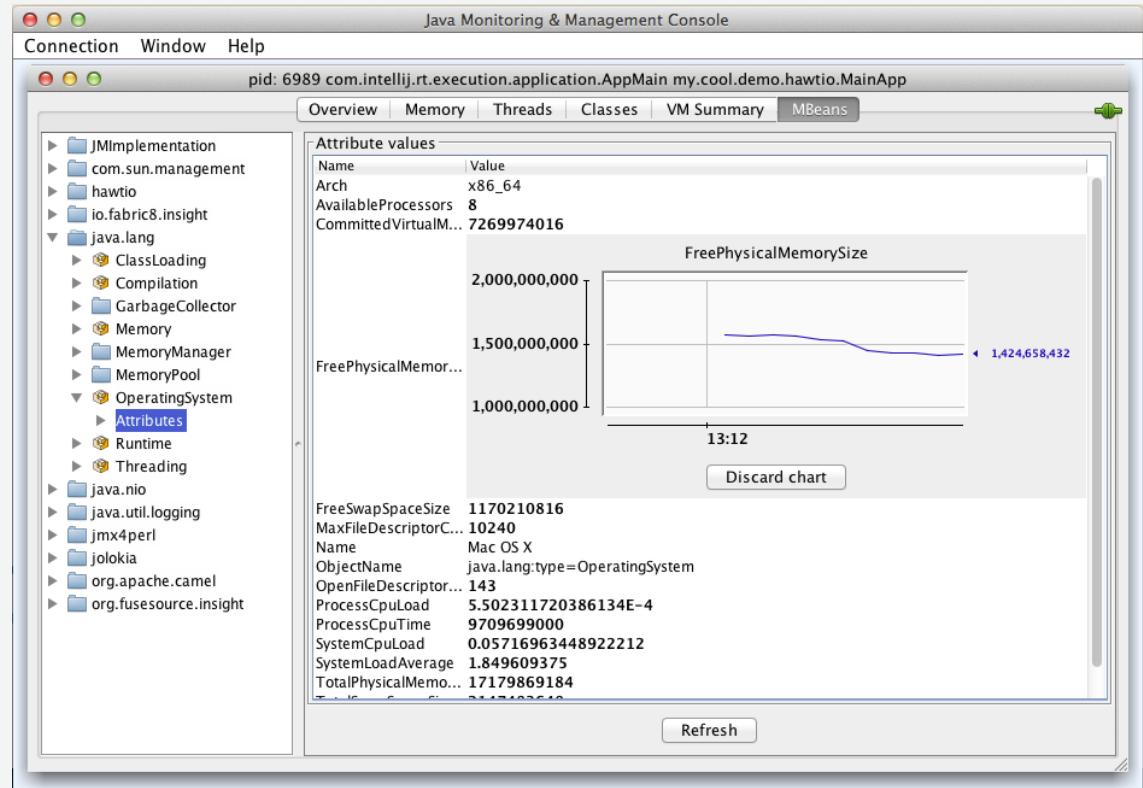
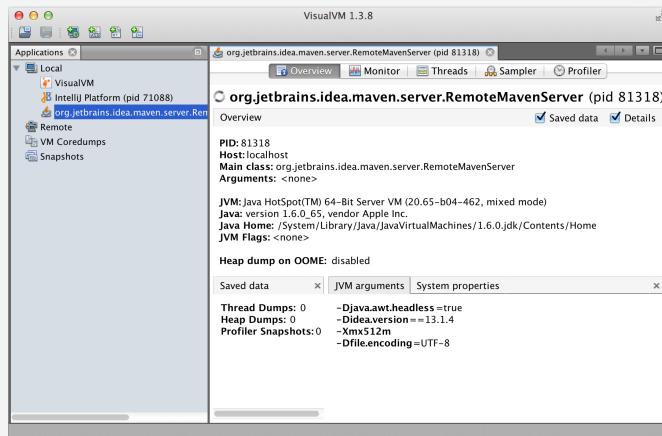
a modular web console for managing your Java stuff

View Demos Get Started Now!



JMX Tools

- JConsole, VisualVM → Developers tool (🔧)
- Proprietary com protocol (RMI/IOP)



- Command line tools → jmxclient, jmxterm, mx4j
- Not so user friendly : 😞

Jolokia ?

- Command line tools → jmx4perl, j4psh
- JMX / HTTP Bridge
- REST API : read attributes, execute operations ☺



A screenshot of a web browser window. The address bar shows the URL: localhost:8282/hawtio/jolokia/read/org.apache.camel:context=camel-1,name="route1",type=routes. The page content displays a JSON object representing the attributes of a Camel route named "route1".

```
{
  - request: {
      mbean: "org.apache.camel:context=camel-1,name=\"route1\",type=routes",
      type: "read"
    },
  - value: {
      StatisticsEnabled: true,
      EndpointUri: "file:///src/test/data?noop=true",
      CamelManagementName: "camel-1",
      ExchangesCompleted: 0,
      LastProcessingTime: 0,
      ExchangesFailed: 0,
      Description: null,
      FirstExchangeCompletedExchangeId: null,
      FirstExchangeCompletedTimestamp: null,
      ExchangeCount: 0,
      ExchangeRate: 0.0
    }
}
```

Modern HTML5 Architecture

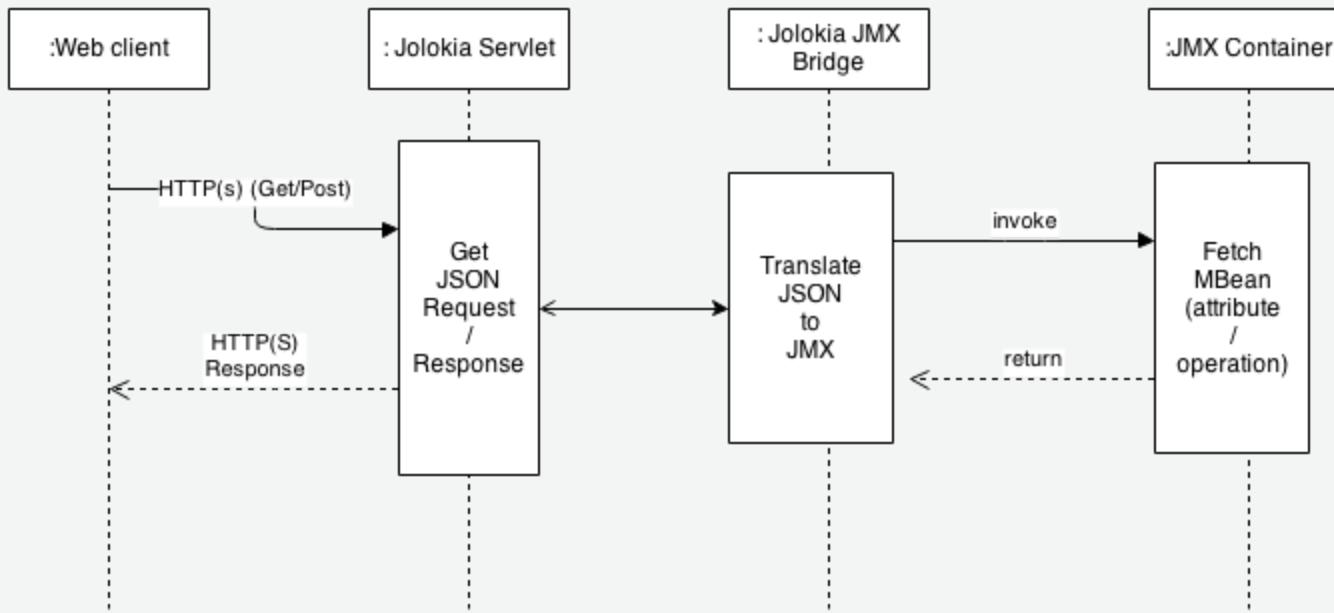
- Hawtio = A combination of the most **powerful web** technologies



- Front end : HTML5 with **AngularJS** & **JSON / REST**
- Backend : Java Servlet & **Jolokia** JMX gateway

Communication ...

- **Web** client communicates to Jolokia **agent** over HTTP(S)
- Send Messages represented in **JSON** Format to Jolokia REST Servlet
- Jolokia **translates** the request/response & map the JSON payload with JMX Calls



Domain

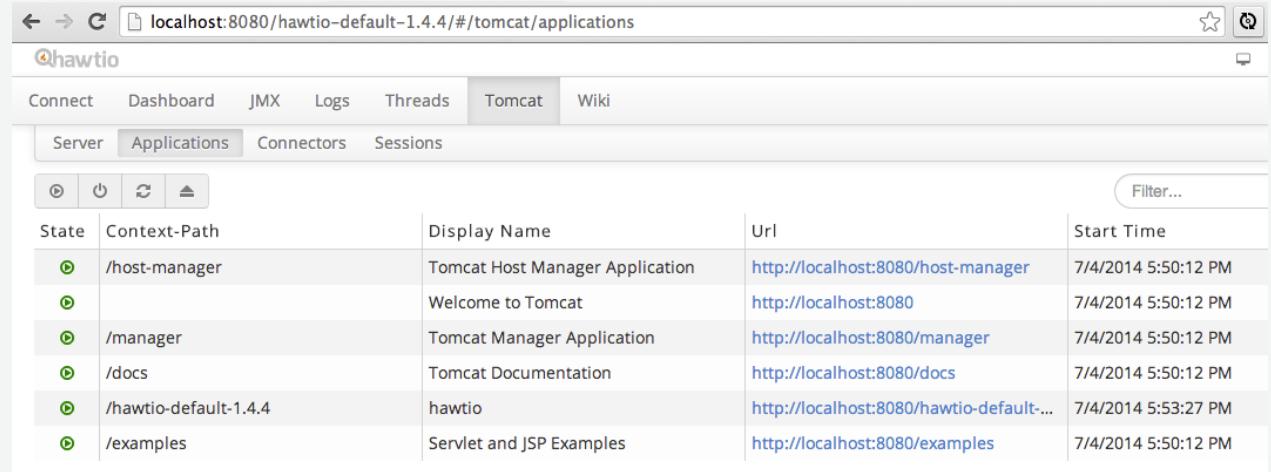
- JMX Domain / context
 - java.lang, com.sun.management, java.util
 - org.apache.camel
 - io.hawt
 - ...
- Attributes & operations

The screenshot shows the Hawtio web interface at `localhost:8080/hawtio-default-1.4.4/#/jmx/operations?nid=root-Catalina`. The top navigation bar includes links for Connect, Dashboard, JMX (selected), Logs, Threads, Tomcat, and Wiki. The left sidebar displays a tree view of the JMX domain structure under the Catalina node, including Cache, Connector, Deployer, Engine, Environment, Filter, GlobalRequestProcessor, Host, JspMonitor, Loader, Manager, Mapper, MBeanFactory, NamingResources, ProtocolHandler, Realm, RequestProcessor, and Resource. To the right of the tree view is a list of available operations, each preceded by a green gear icon:

- reload()
- removeApplicationListener(java.lang.Object)
- removeApplicationParameter(java.lang.String)
- removeChild(java.lang.String)
- removeInstanceListener(java.lang.String)
- removeLifecycleListeners(java.lang.String)
- removeMimeTypeMapping(java.lang.String)
- removeParameter(java.lang.String)
- removeRoleMapping(java.lang.String)
- removeSecurityRole(java.lang.String)

Plugin

- What is a **plugin** ?
- →
- Collection of UI
- JSON request(s)/response(s)
- JS lib to do the rendering



The screenshot shows a web browser window with the URL `localhost:8080/hawtio-default-1.4.4/#/tomcat/applications`. The page is titled "hawtio". The navigation bar includes links for Connect, Dashboard, JMX, Logs, Threads, Tomcat, and Wiki. The "Applications" tab is selected. Below the tabs is a toolbar with icons for refresh, stop, start, and filter. A "Filter..." input field is also present. The main content is a table listing Tomcat applications:

State	Context-Path	Display Name	Url	Start Time
●	/host-manager	Tomcat Host Manager Application	http://localhost:8080/host-manager	7/4/2014 5:50:12 PM
●	/	Welcome to Tomcat	http://localhost:8080	7/4/2014 5:50:12 PM
●	/manager	Tomcat Manager Application	http://localhost:8080/manager	7/4/2014 5:50:12 PM
●	/docs	Tomcat Documentation	http://localhost:8080/docs	7/4/2014 5:50:12 PM
●	/hawtio-default-1.4.4	hawtio	http://localhost:8080/hawtio-default-1.4.4	7/4/2014 5:53:27 PM
●	/examples	Servlet and JSP Examples	http://localhost:8080/examples	7/4/2014 5:50:12 PM

Plugin Front

- Mix of HTML tags & angular directives **ng-***

```
<div class="row-fluid" ng-controller="Tomcat.TomcatController"> ①
  <div class="row-fluid">
    <div class="pull-left">
      <form class="form-inline no-bottom-margin">
        <fieldset>
          <div class="controls control-group inline-block controls-row">
            <div class="btn-group">
              <button ng-disabled="selected.length == 0" class="btn" ng-click="start()" title="Start"
                      class="icon-play-circle"></i></button>
              <button ng-disabled="selected.length == 0" class="btn" ng-click="stop()" title="Stop"
                      class="icon-off"></i></button>
              <button ng-disabled="selected.length == 0" class="btn" ng-click="reload()" title="Reload"
                      class="icon-refresh"></i></button>
              <button ng-disabled="selected.length == 0" class="btn" ng-click="uninstallDialog.open()"
                      class="icon-eject"></i></button>
            </div>
          </div>
        </fieldset>
      </form>
```

①

Angular ng-controller defined within HTML <div> tag

Plugin Controller

```
module Tomcat {
  var pluginName = 'tomcat';
  export var _module = angular.module(pluginName, ['bootstrap', 'ngResource', 'ui.bootstrap.dialog'])

  _module.config(['$routeProvider', ($routeProvider) => {
    $routeProvider.
      when('/tomcat/server', {templateUrl: 'app/tomcat/html/server.html'}).
      when('/tomcat/apps', {templateUrl: 'app/tomcat/html/apps.html'}). ①
      when('/tomcat/connectors', {templateUrl: 'app/tomcat/html/connectors.html'}). ②
      when('/tomcat/sessions', {templateUrl: 'app/tomcat/html/sessions.html'});
  }]);
}
```

-
- ① Map HTTP request with applications HTML page
 - ② Idem for the connectors page

Jolokia Request

- Javascript function called to execute a GET request of type **exec** or **read**
- Mbean & Attributes OR Operation are passed as parameters
- **onSuccess** → calls function for the rendering

```
jolokia.request({  
    type: 'exec',  
    mbean: id,  
    operation: op,  
    arguments: null  
},  
    onSuccess($scope.onResponse, {error: $scope.onResponse}));
```

Jolokia Search

- Search operation is supported
- Find MBeans according to a search **query** based on the type of the MBean to find

```
function loadData() {  
    var connectors = jolokia.search("*:type=Connector,*"); ①  
    if (connectors) {  
        var found = false;  
        angular.forEach(connectors, function (key, value) {  
            var mbean = key;  
            if (!found) {  
                var data = jolokia.request({type: "read", mbean: mbean, attribute: ["port", "scheme", "prot  
...  
        jolokia.search("*:j2eeType=WebModule,*", onSuccess(render)); ②  
    }  
}
```

① A search query to find the type Connector

② Anoter search query for mbeans based on the type
" * :j2eeType=WebModule, * "

Jolokia Response

```
function render(response) { ①
    response = Tomcat.filerTomcatOrCatalina(response);

    $scope.webapps = [];
    $scope.mbeanIndex = {};
    $scope.selected.length = 0;

    function onAttributes(response) {
        var obj = response.value;
        if (obj) {
            obj.mbean = response.request.mbean; ②
            var mbean = obj.mbean;

            // compute the url for the webapp, and we want to use http as scheme
            var hostname = Core.extractTargetUrl($location, $scope.httpScheme, $scope.httpPort);
            obj.url = hostname + obj['path'];
        }
    }
}
```

① Response rendered & parsed

② JSON result mapped with angular objects **\$scope.***

Plugins

- **Plug-an-play** architecture (> 25 plugins)
- UI updated in **real time**
- Some are **Packaged** : jvm, threads, dashboard, camel, activemq, ...
- Some are **Server side** : git, maven, aether, log
- Some are **External** : insight, elasticsearch, kibana
- **Reusable** for developers (branding, datatable, forms, ide, perspective, tree, ui)

What Front looks like

The screenshot shows the hawtio welcome page at localhost:9090/hawtio/#/welcome. The top navigation bar includes links for Container, Connect, Dashboard, JMX, Logs, Threads, and Wiki. A red arrow points to the 'Wiki' link. The main content area features a 'Welcome to Qhawtio' header and a brief introduction about hawtio being a lightweight and modular HTML5 web console with lots of plugins for managing Java stuff. It also provides sections for General Navigation, Switching Perspectives, Getting Help, and Logging Console, along with a Preferences link.

Top level bar with menus

Container Connect Dashboard JMX Logs Threads Wiki

Welcome to Qhawtio

JS Co

General Navigation

Primary navigation in hawtio is via the top navigation bar.

Clicking on a navigation link will take you to that plugin's main page.

Switching Perspectives

You can switch between perspectives in hawtio by clicking the (▼) perspective menu, which is the left-most item in the main navigation bar. This menu also maintains the last 5 remote connections that have been made so that you can quickly connect to other JVMs. If there is no connection currently, then the (▼) perspective menu may not be shown.

Getting Help

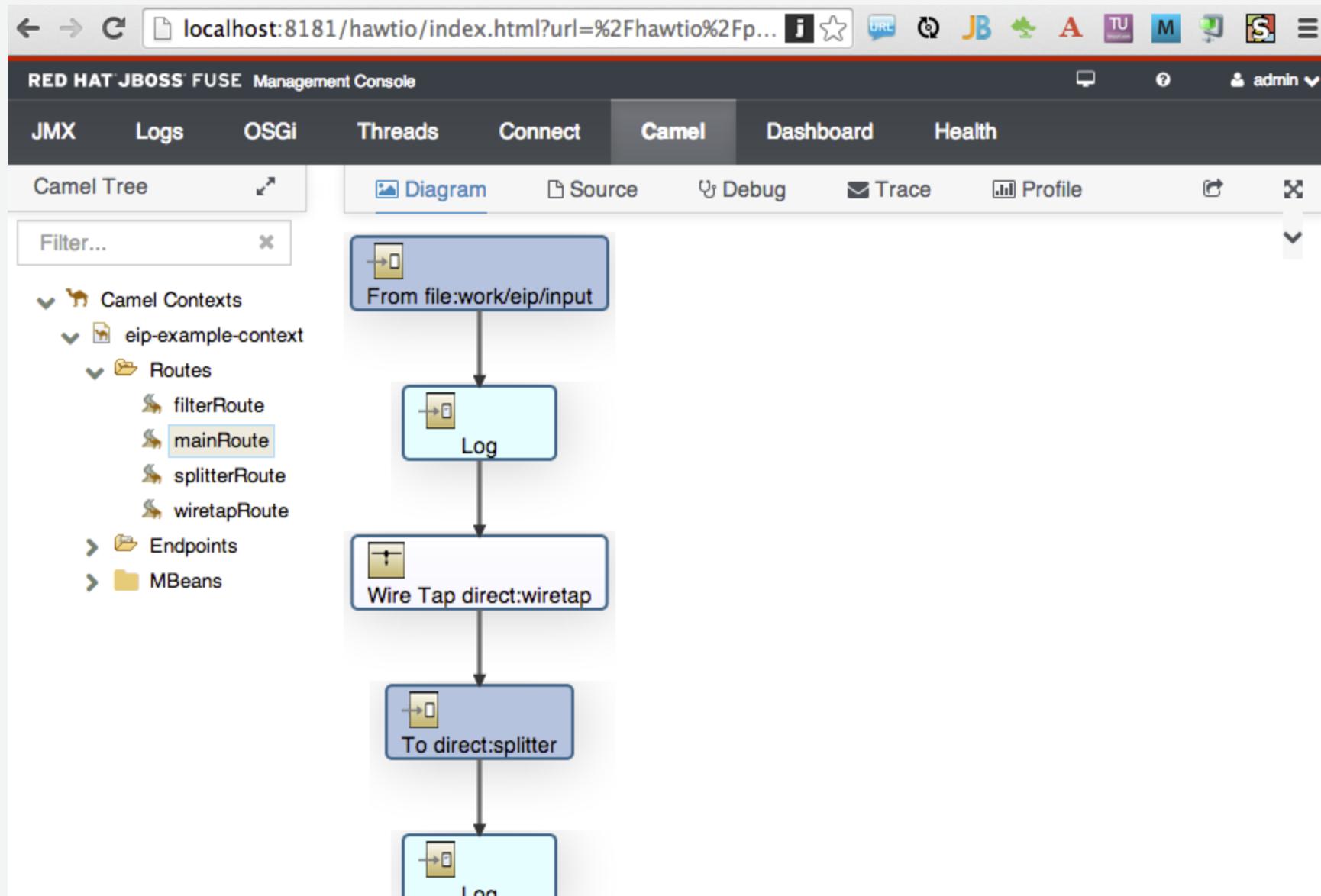
Click the Help icon (ⓘ) in the main navigation bar to access hawtio's help system. Browse the available help topics for plugin-specific help navigation bar on the left.

Logging Console

The logging console is accessible by clicking the icon (terminal) in the main navigation bar. Information from the console can be useful when reporting bugs to the hawtio community. And from the Preferences you can configure the logging level used by the console logger.

Preferences

Discover your camel routes

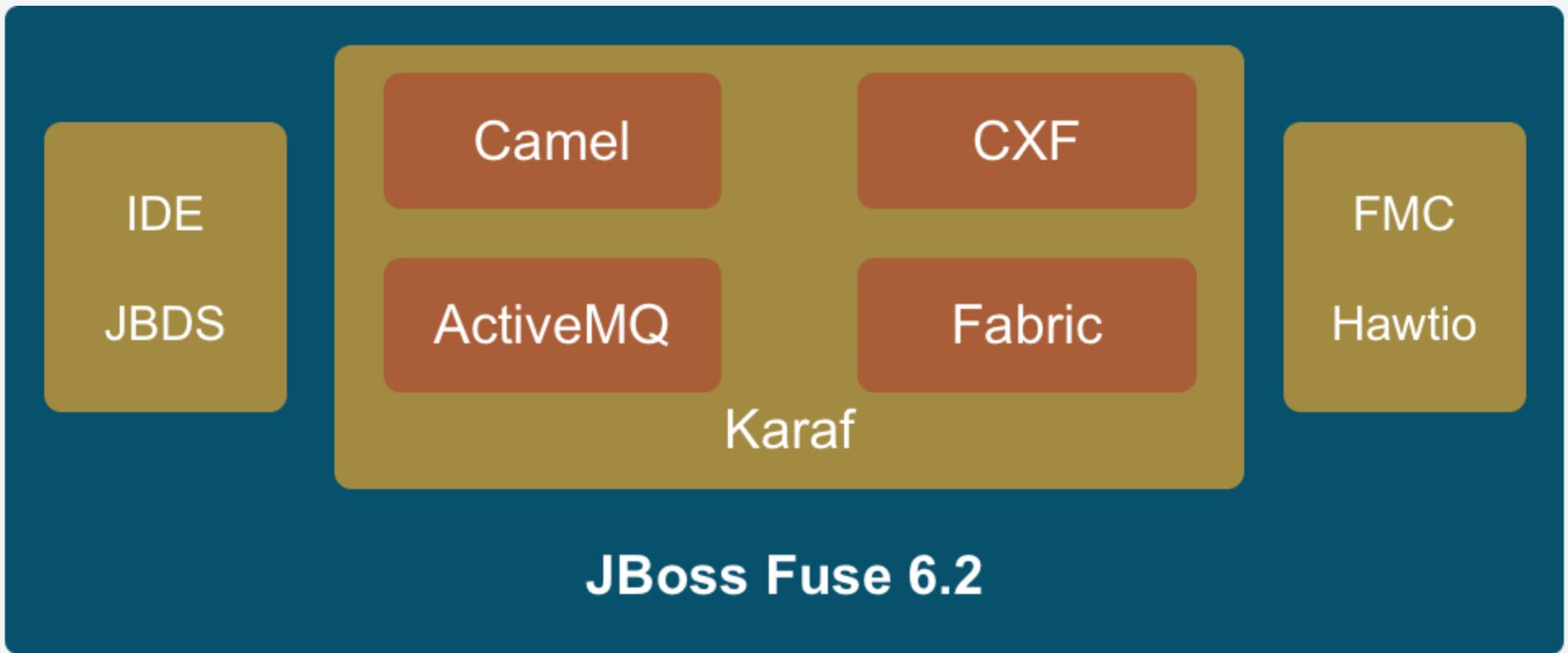


Manage your brokers

The screenshot shows the Red Hat JBoss FUSE Management Console interface for ActiveMQ. The top navigation bar includes links for ActiveMQ, JMX, Logs, OSGi, Terminal, Threads, and Connect. The left sidebar is titled "ActiveMQ Tree" and shows a tree structure with "root" expanded, revealing "Queue", "Topic", "clientConnectors", "Health", and "PersistenceAdapter". The main content area is titled "Browse" and displays a table of messages in the "Queue" folder. The table columns are: Message ID, Type, Pri..., Timestamp, Expires, Repl..., and Corre... . There are five rows of message data, each starting with "ID:Dabou.local-64432-13987..." and showing a timestamp of "2014-04-29T1..." and a priority of "0".

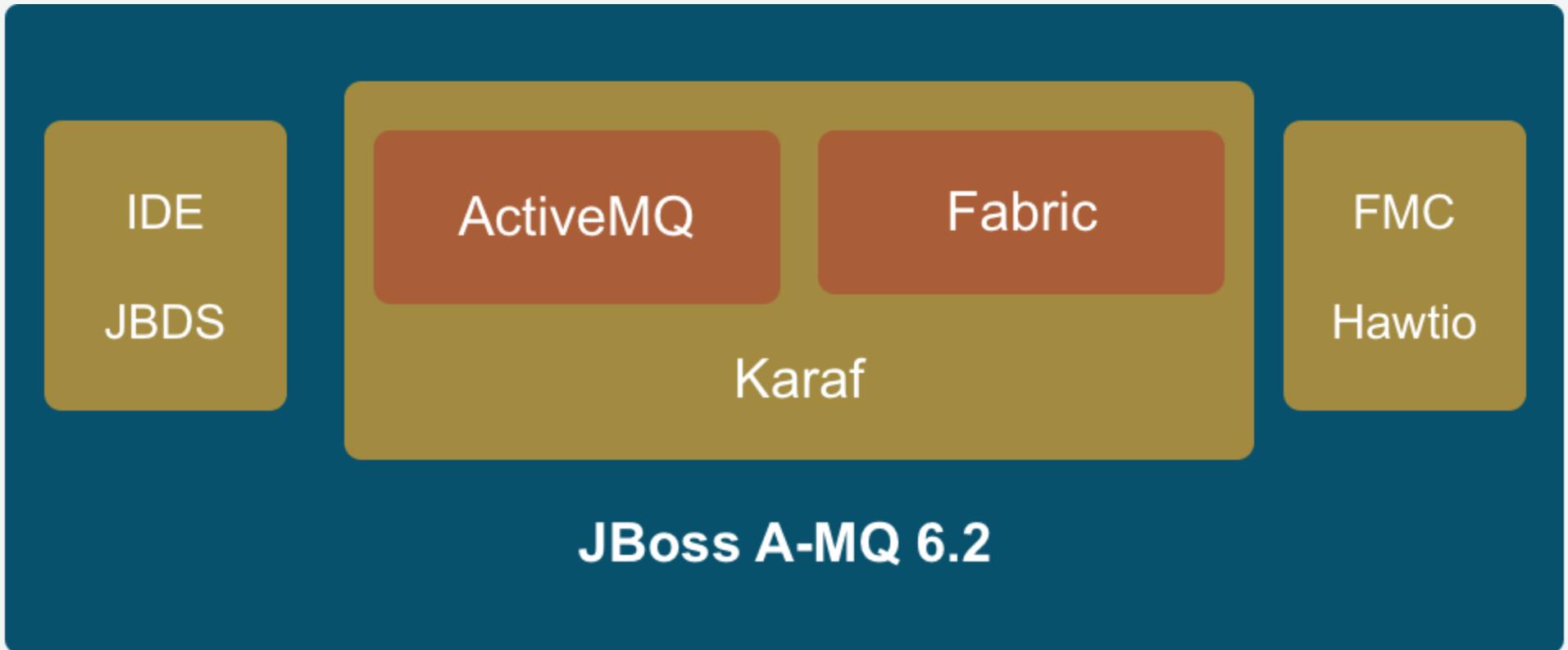
The screenshot shows the Red Hat JBoss FUSE Management Console interface for the Fabric view. The top navigation bar includes links for Runtime, Wiki, Dashboard, and Health. The main menu bar has tabs for Containers, Profiles, Manage, MQ (which is currently selected), APIs, EIPs, and Registry. Below the menu, there is a search bar labeled "Filter..." and buttons for "+ Broker" and "+ Container". The central area displays a diagram of the fabric structure. It shows two containers: "default" (represented by a white box) and "jboss-fuse-full" (represented by a blue box). The "jboss-fuse-full" container is expanded, showing a "broker" node with a green circular icon containing a white arrow pointing right. At the bottom right of the interface, there is a status bar with the text "root pid:13919".

Integration & Mediation Platform



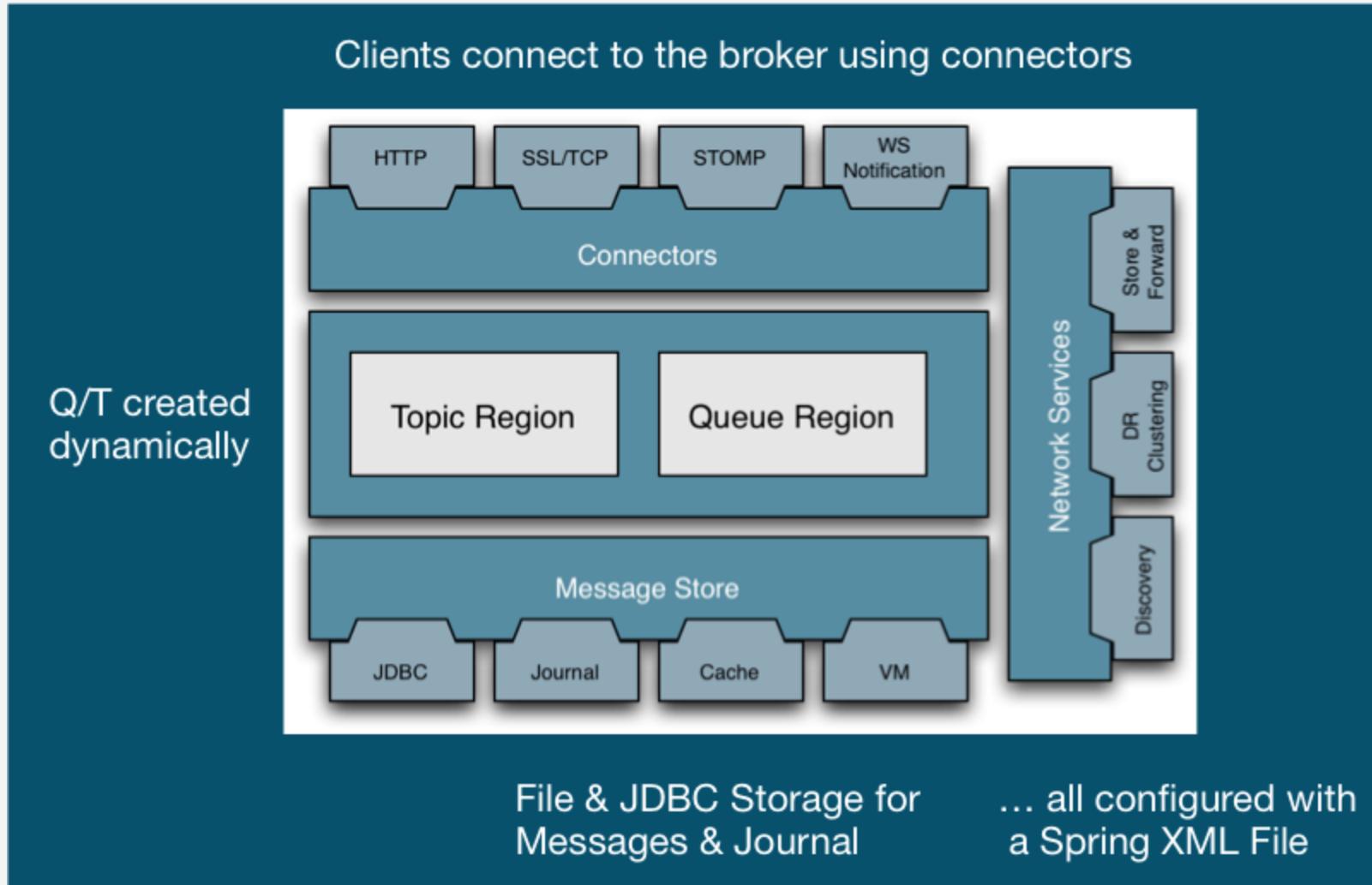
- **> 100** components : file, jms, ftp, ...
- **> 50** EIP Patterns : content based router, splitter, aggregator, ...

Messaging Broker Platform



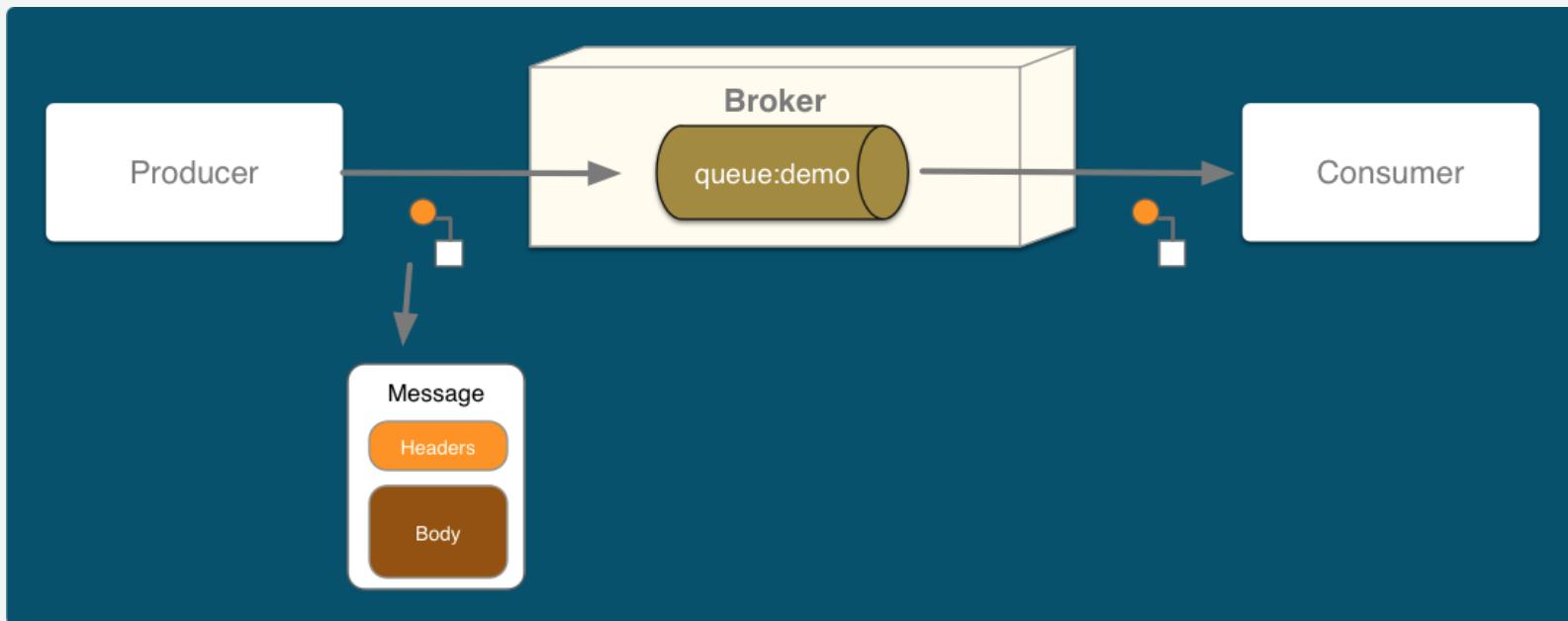
- Support : JMS, AMQP, MQTT, STOMP, Websocket
- **iPaaS** (Openshift cartridges - v2)

Broker Architecture



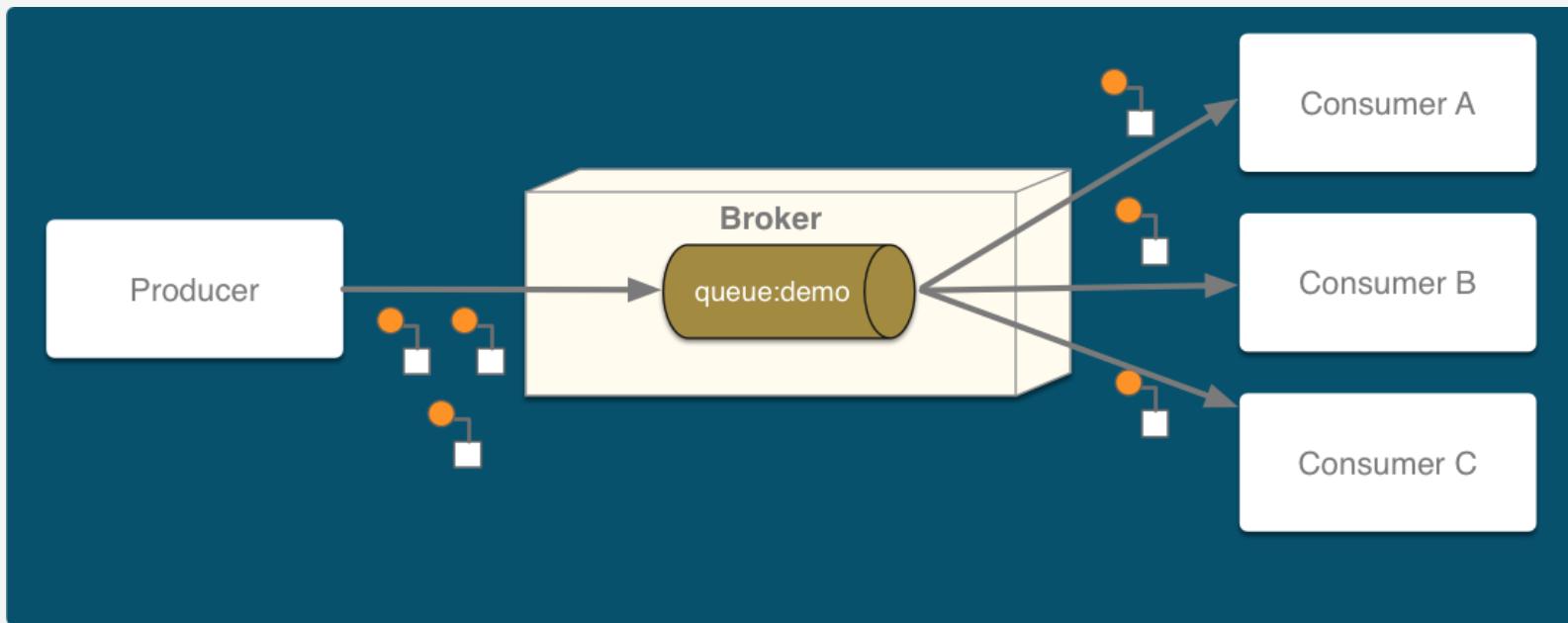
Point-to-Point Messaging

- Producer sends message(s) to queue using JMS API
- Consumer listens for message(s) from queue
- Messages stored until read or expired
- Messages can be persisted, are read only once



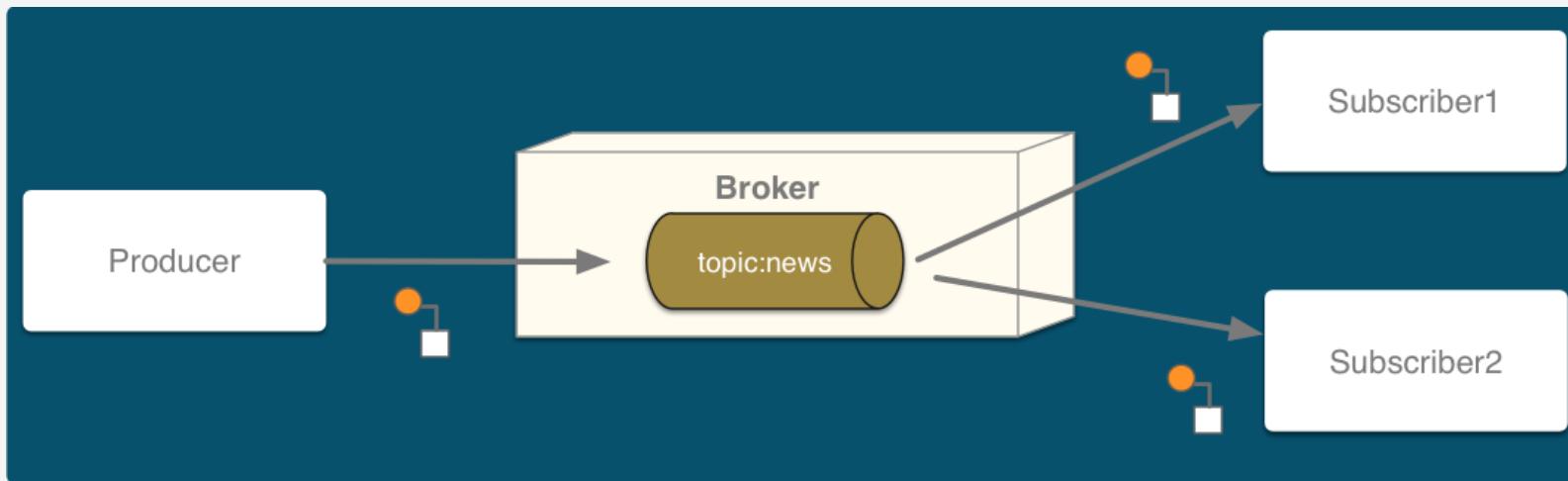
Load Balancing

- Workload can be distributed between connected clients
- Round-robin algorithm used



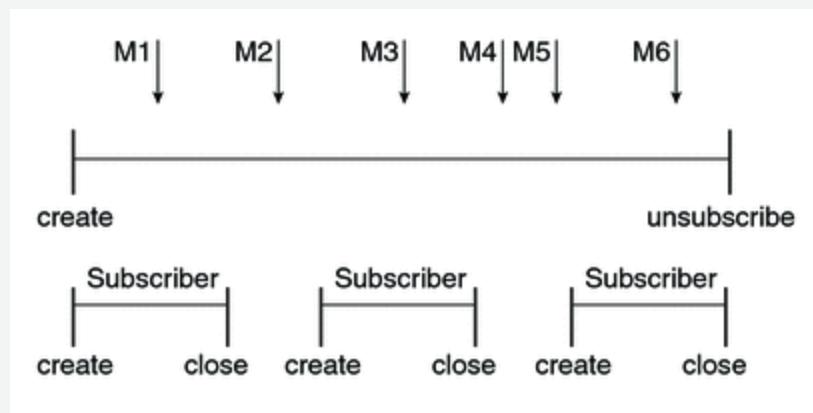
Publish and Subscribe

- Client sends message to topic
- Broker sends message to all currently connected subscribers
- Messages are consumed x times (1-to-many)



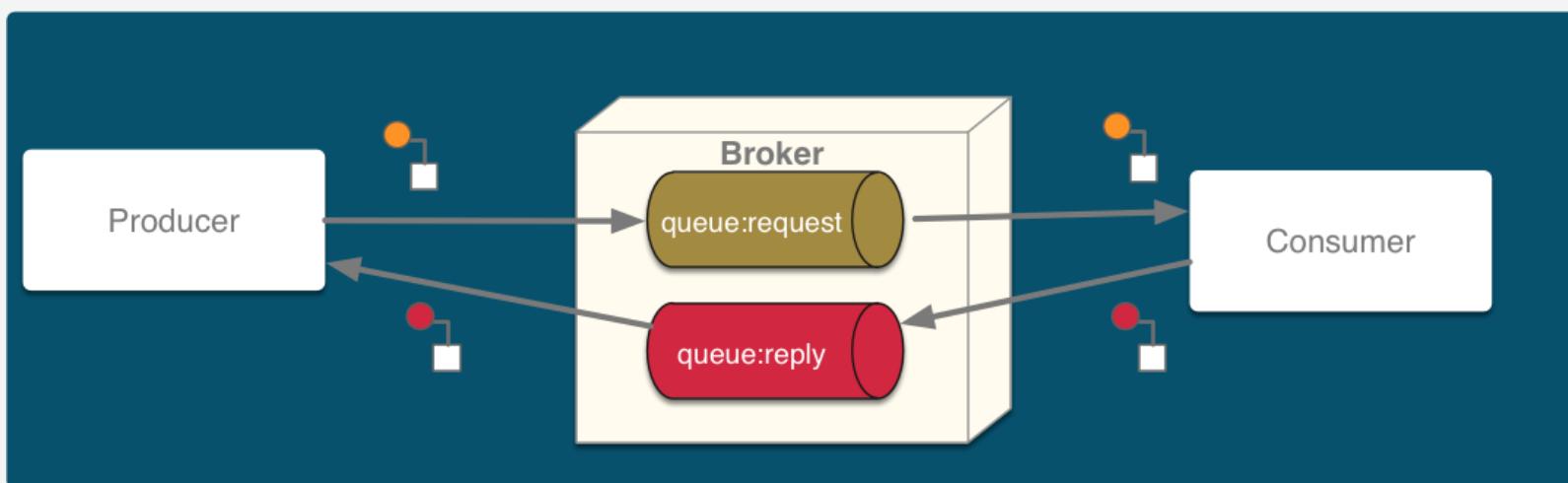
Durable Subscriber

- Messages usually expire when subscriber(s) not connected
- Supports connection/disconnection
- `DurableSubscriber` property = unique client ID registered to broker
- Subscriber client ID controls messages delivered to broker



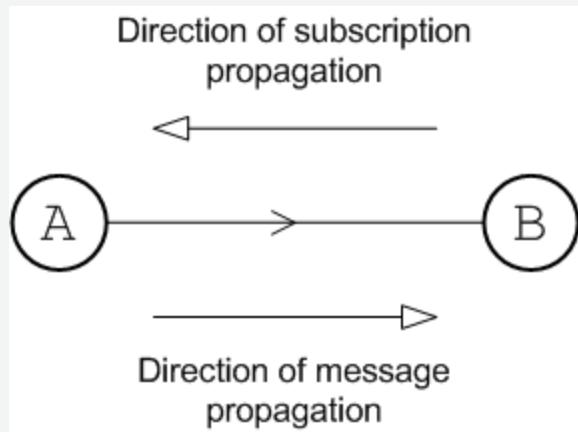
Request/Reply

- Producer sets 2 properties to send message
 - `JMSReplyTo`: Destination for response message
 - `JMSCorrelationID`: Unique ID matches/correlates messages (request/response)
- Consumer replies to `JMSReplyTo` destination using `JMSCorrelationID` in message

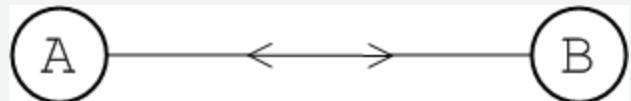


Network of Brokers

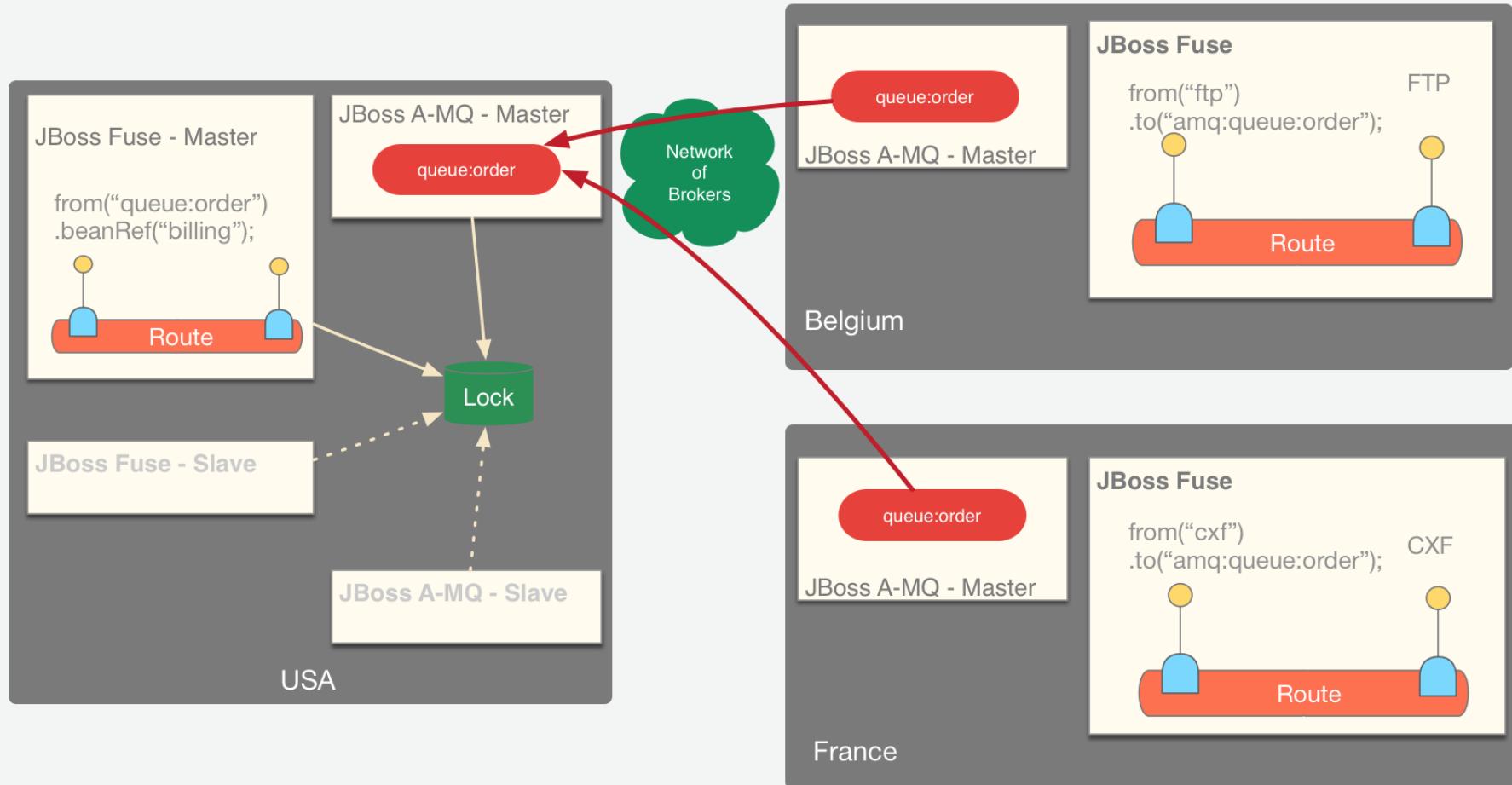
- Created when one broker establishes network connection to another broker:



- Supports multiple connectors
- To define as bi-directional, use `duplex=true`:



Big picture



Future - Fuse 7

- Fabric8 - V2
- More integrated → Openshift v3, Kubernetes & Docker
- Service Management & Governance : ApiMan
- OAuth2 Server, IDM : KeyCloak

JBoss Fuse in action

- **DEMO**

- Design REST Service using **Camel**
- Deploy on **JBoss Fuse** using a **profile**
- Create container & expose REST Service using **Fabric**
- Define **Security Policy** for the endpoint →
 - **Authenticate** the user (**BASIC** - static realm)
 - **Authorize** using **Oauth2**

Rest DSL in Action : <https://github.com/FuseByExample/rest-dsl-in-action>

Camel & Mobile : <https://github.com/cmoulliard/feedhenry-camel>

Questions



- Twitter : [@cmoulliard](https://twitter.com/cmoulliard)

- More info →
 - www.jboss.org/products/fuse.html
 - <http://www.redhat.com/en/technologies/jboss-middleware>