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

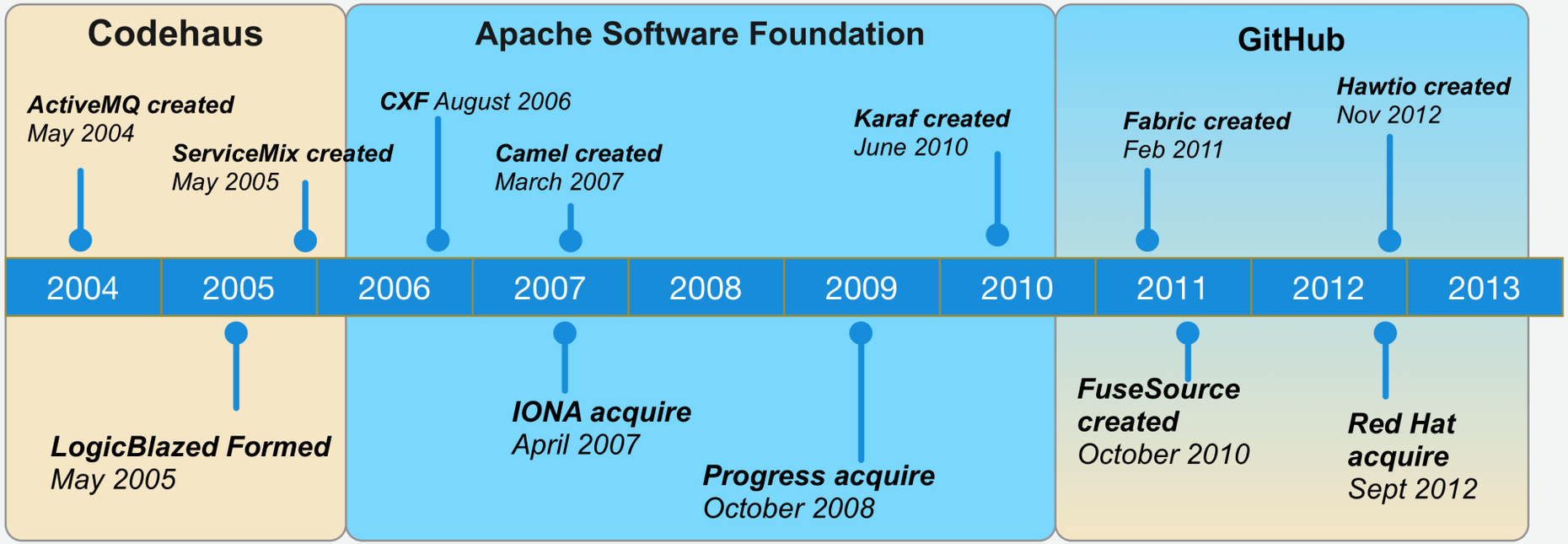
Blog: http://cmoulliard.github.io

Twitter: @cmoulliard

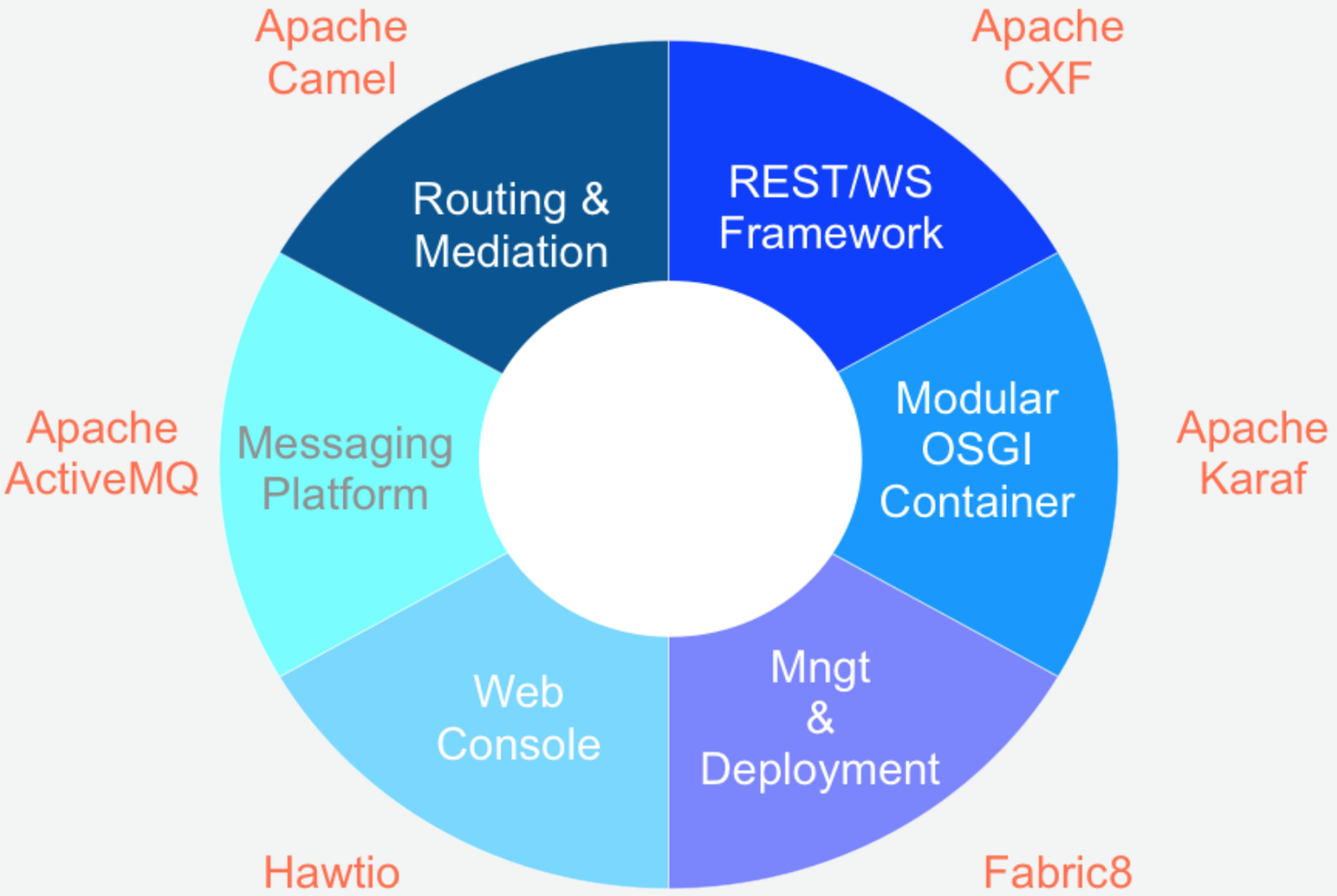
Email: cmoulliard@redhat.com

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

Long **History** behind **Fuse**

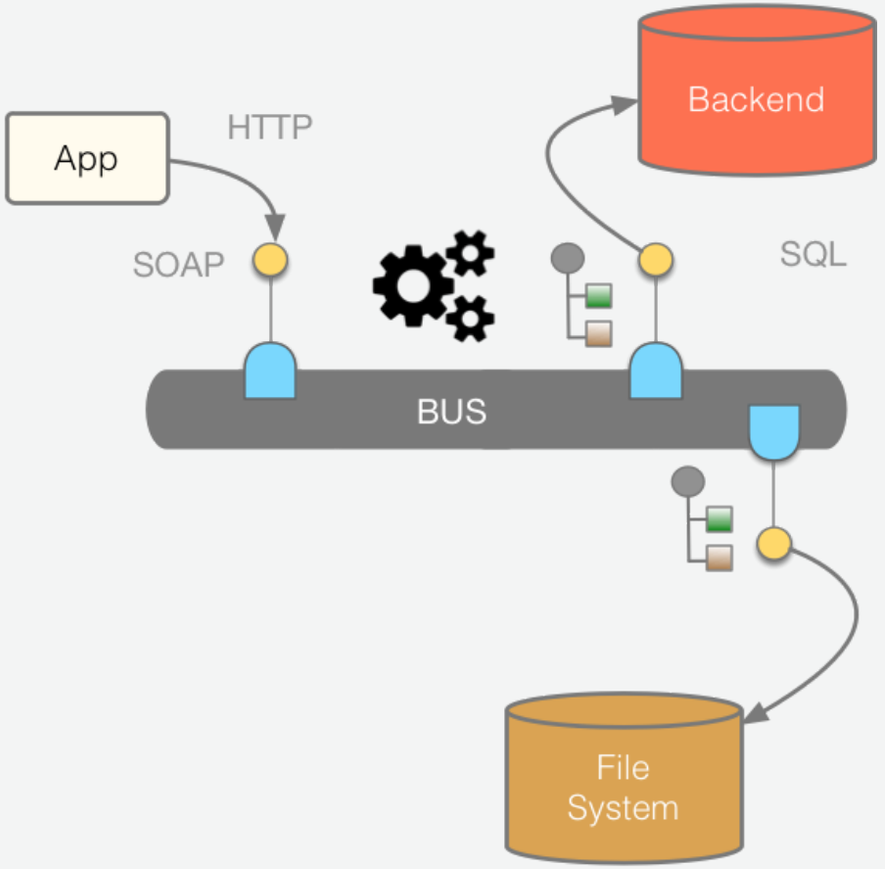


Fuse **Integration** Technology



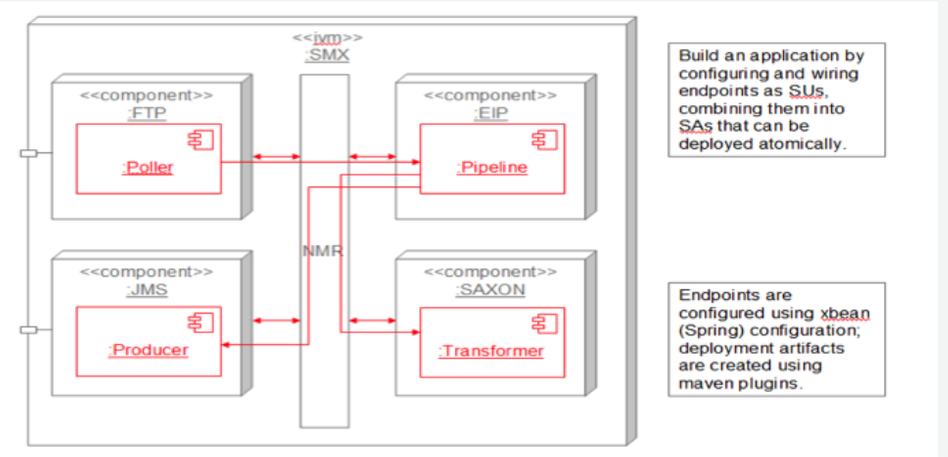
Integration platform

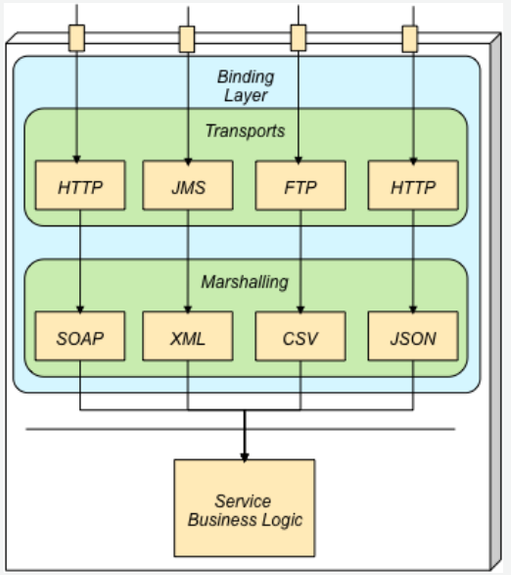
- ⚙ **Manage Complex use cases** ➔ correlation, orchestration, routing, mediation, transformation
- 🔧 Provide **BUS** ➔ exchange messages
- ⚙ Designed around different specs: **JBI**, **SCA**



ESB & JBI

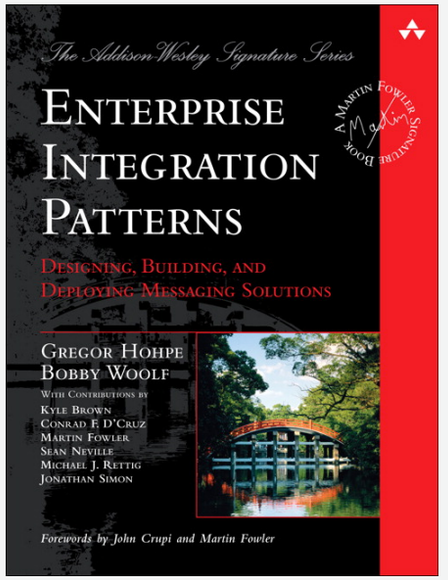
- Services & Components communicate using **NMR** bus
- Packaged as zip (=SU) in a big zip (=SA) including xml config, target service
- Constraint : messages formated as **XML**
- ❗ EIP not included in the spec





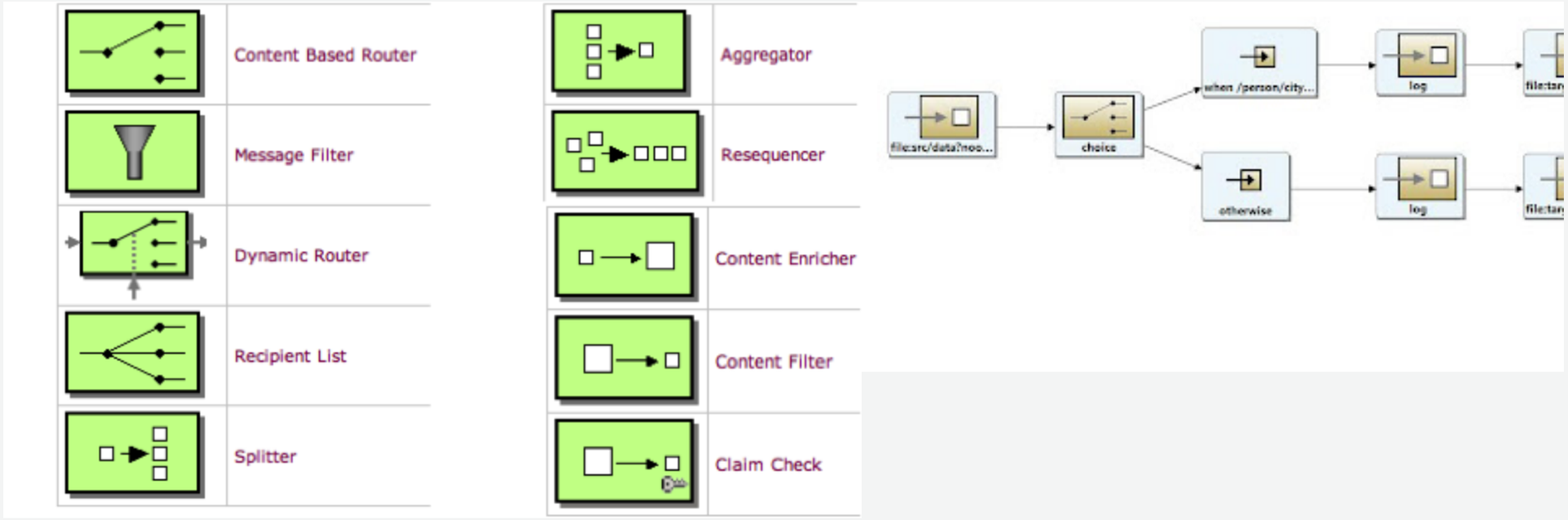
Apache Camel

- Java **Integration** Framework
- Implements ➔ **D**omain **S**pecific **L**anguage
- Supports **E**nterprise **I**ntegration **P**atterns



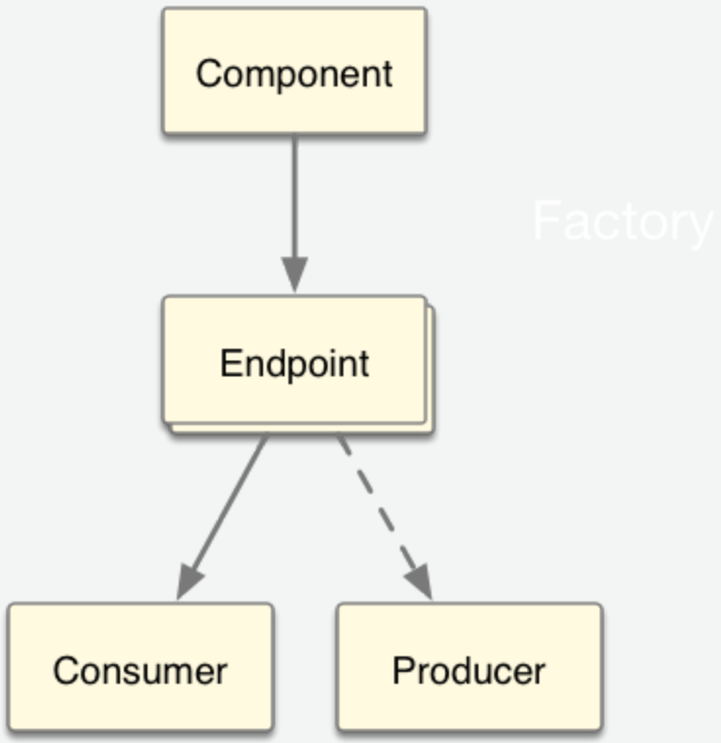
Enterprise Patterns

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



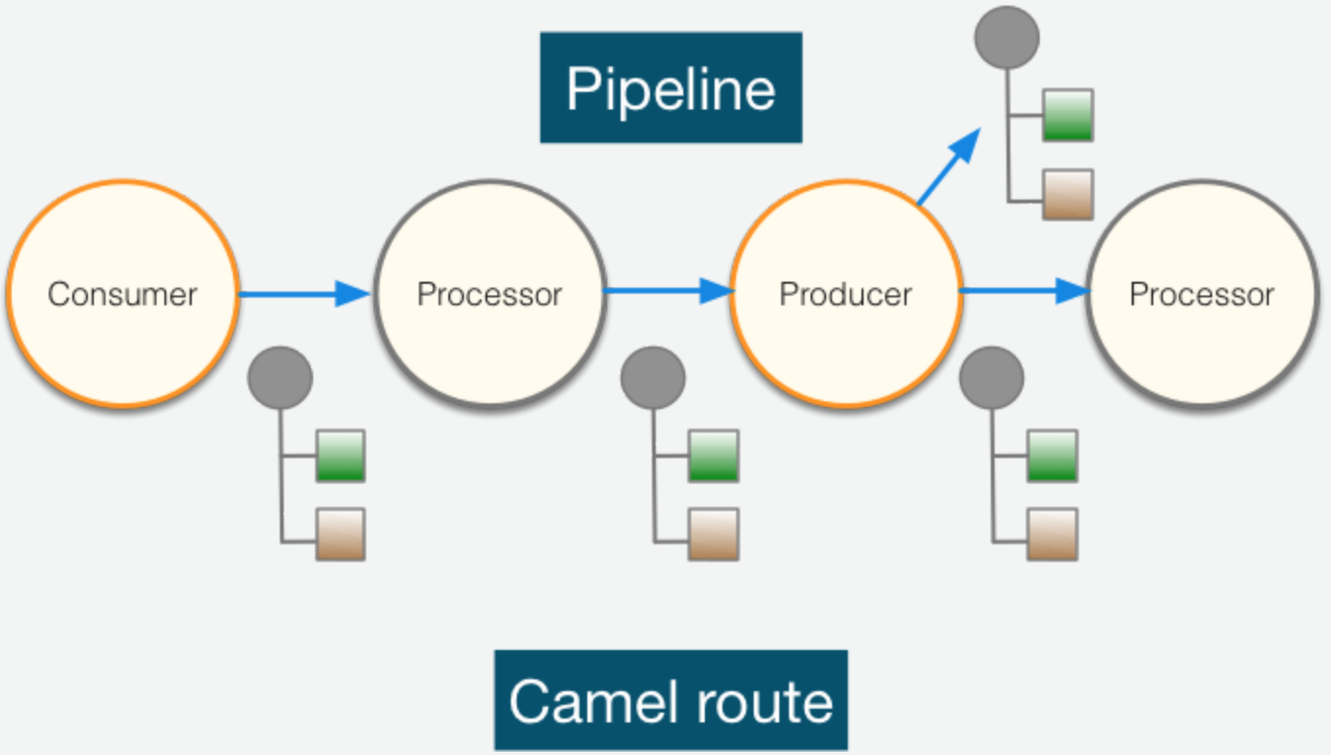
Key concepts

- Component
- Endpoint
- Consumer
- Producer



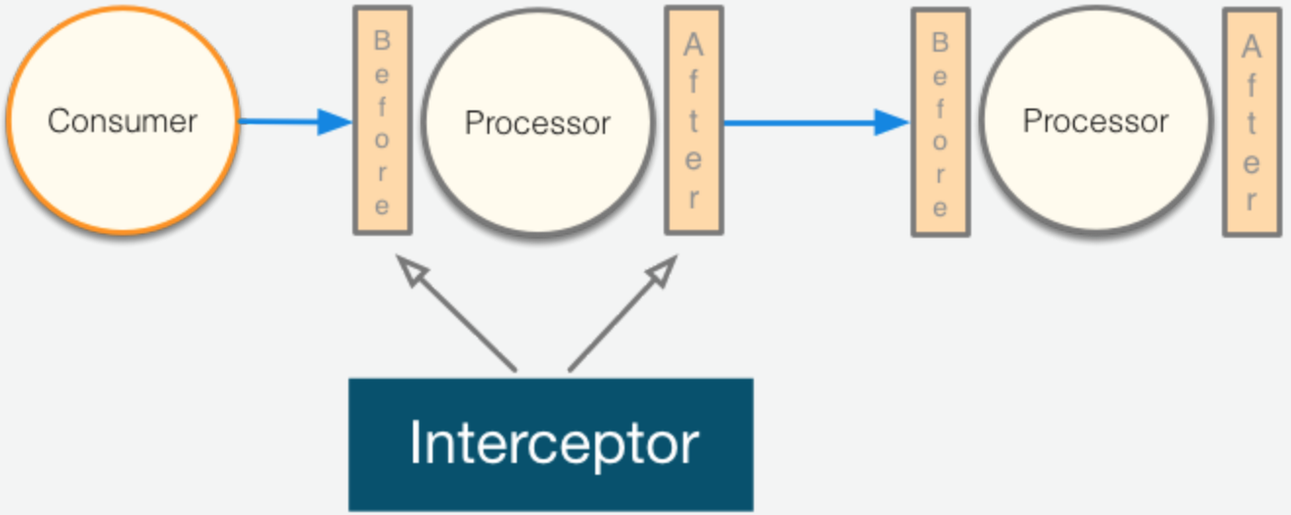
Route, processor

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



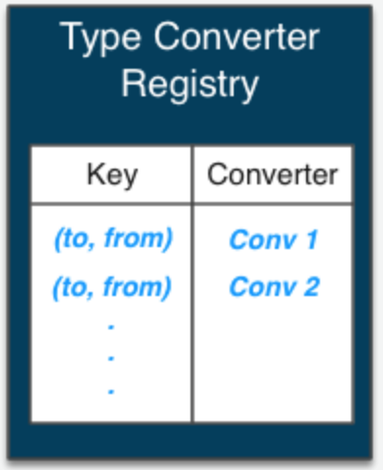
Interceptor

- 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



Data Format

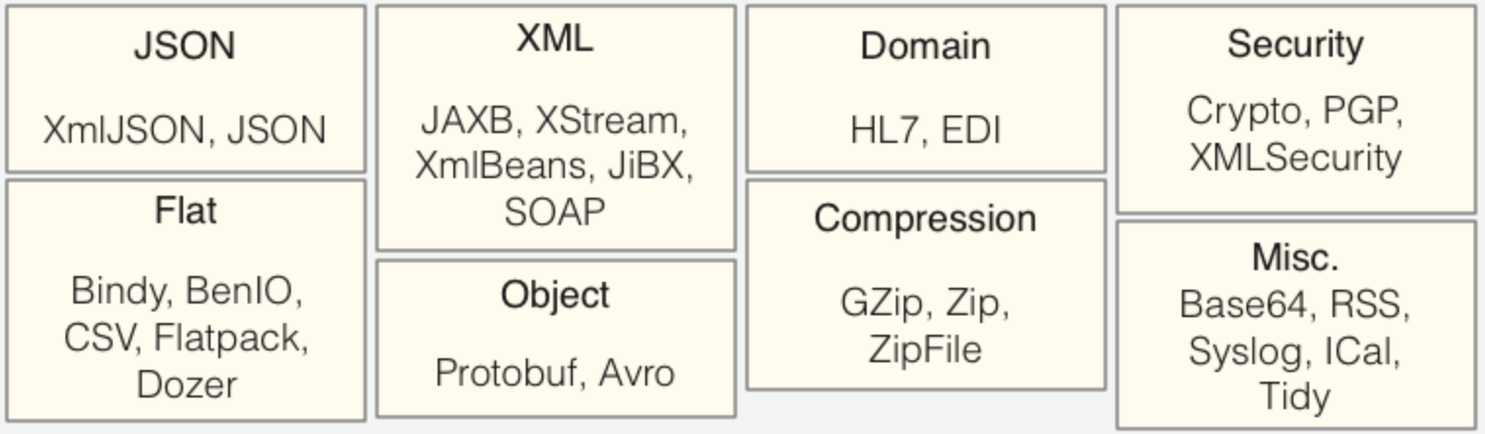
- **Data Transformation** for complex use case

```
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;
}
```

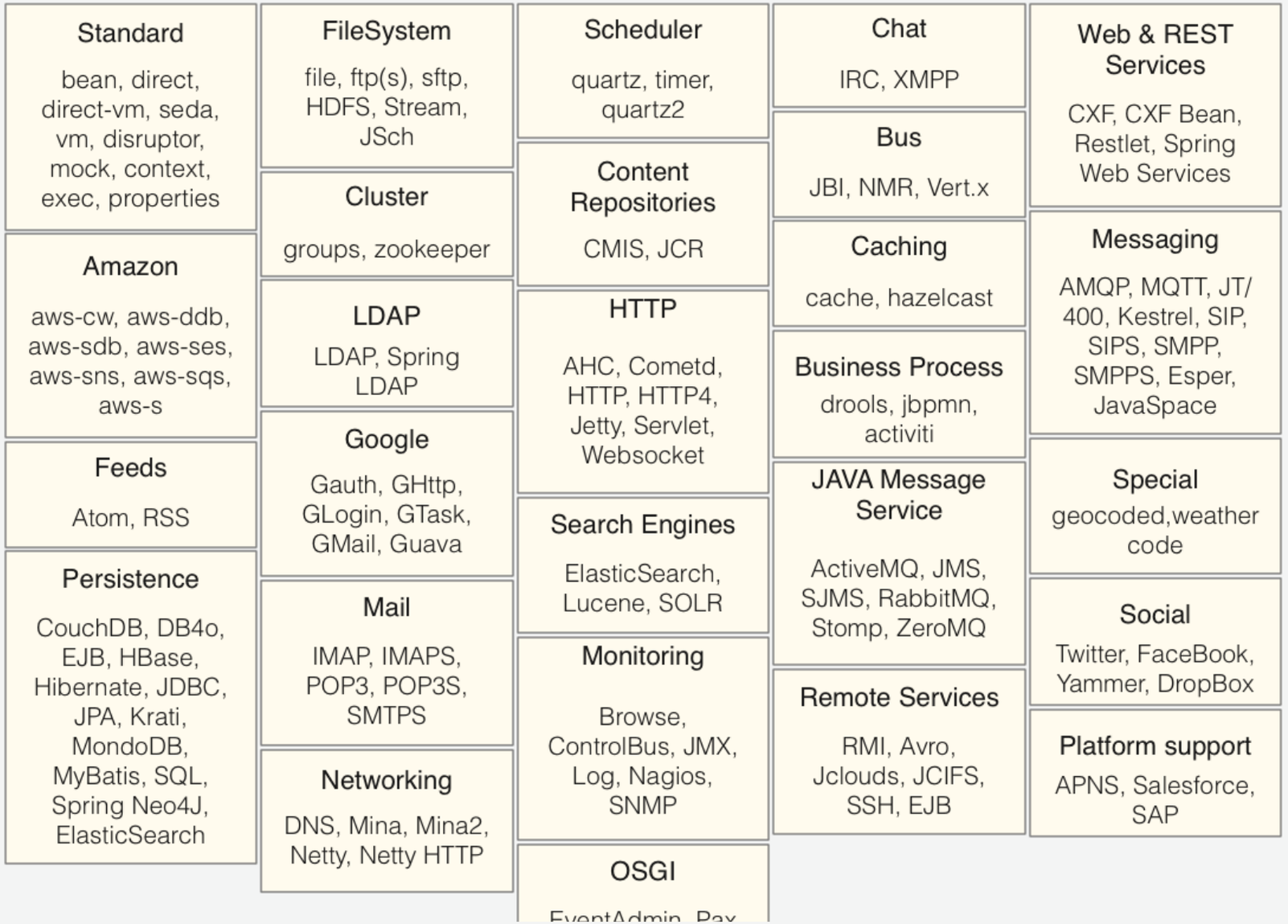
- **Marshalling** : Object ➔ XML (JAXB)
- **Unmarshalling** : XML ➔ Object (JAXB)

Data Format supported

- > 20 Data Format

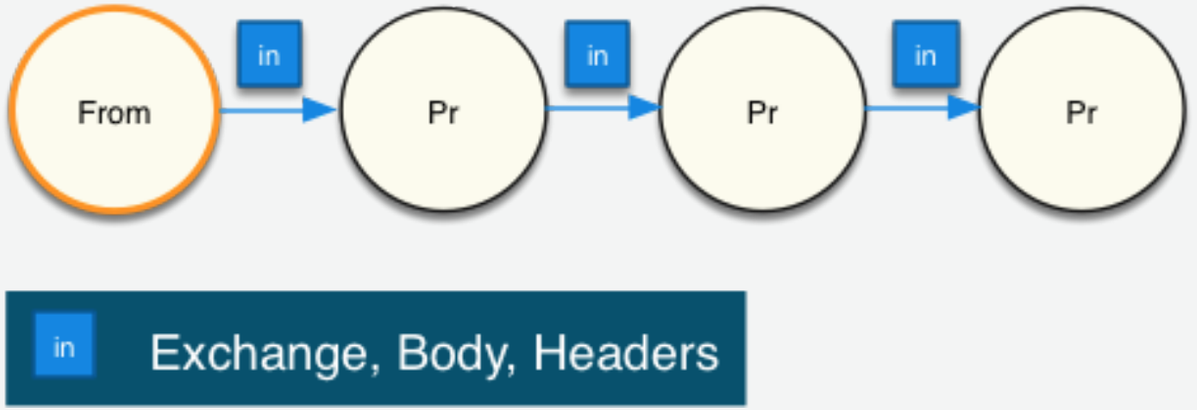


Components

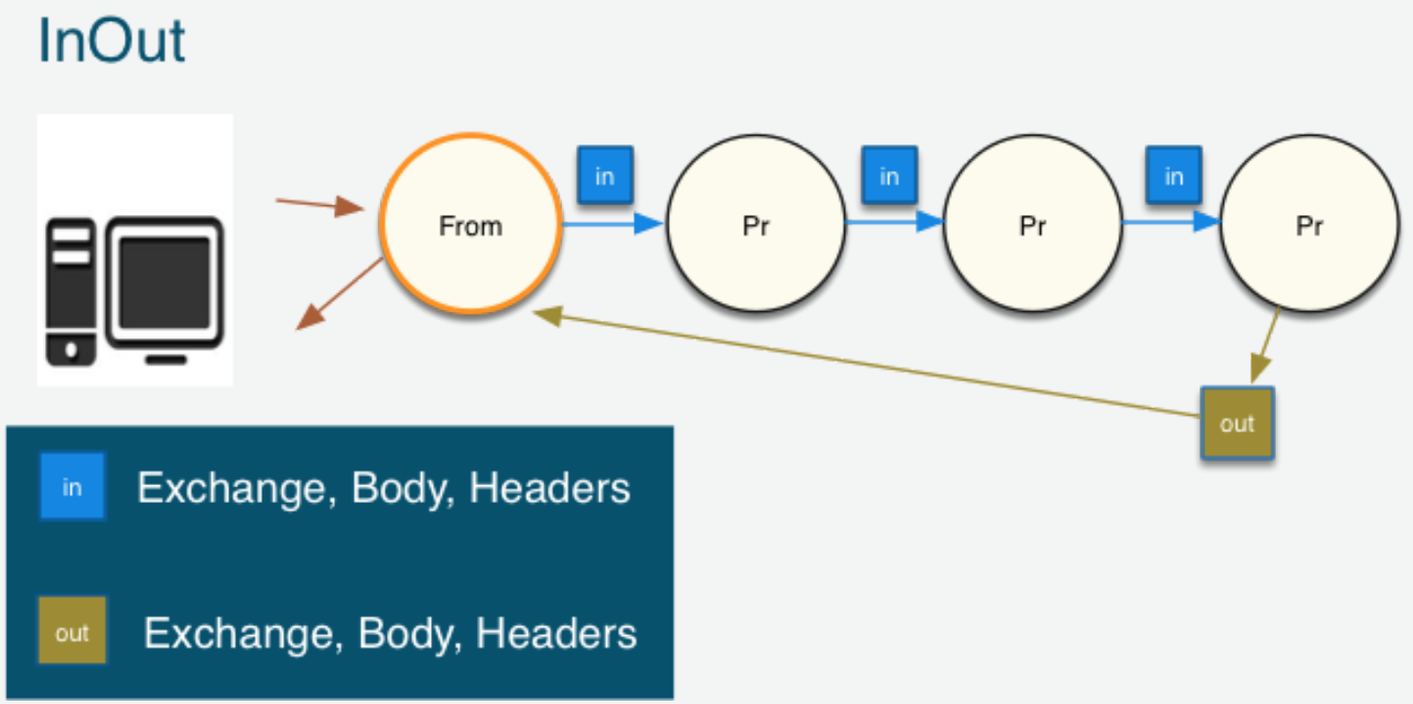

Standard
bean, direct, direct-vm, seda, vm, disruptor, mock, context, exec, properties
Amazon
aws-cw, aws-ddb, aws-sdb, aws-ses, aws-sns, aws-sqs, aws-s
Feeds
Atom, RSS
Persistence
CouchDB, DB4o, EJB, HBase, Hibernate, JDBC, JPA, Krati, MondoDB, MyBatis, SQL, Spring Neo4J, ElasticSearch
FileSystem
file, ftp(s), sftp, HDFS, Stream, JSch
Cluster
groups, zookeeper
LDAP
LDAP, Spring LDAP
Google
Gauth, GHttp, GLogin, GTask, GMail, Guava
Mail
IMAP, IMAPS, POP3, POP3S, SMTPS
Networking
DNS, Mina, Mina2, Netty, Netty HTTP
Scheduler
quartz, timer, quartz2
Content Repositories
CMIS, JCR
HTTP
AHC, Cometd, HTTP, HTTP4, Jetty, Servlet, Websocket
Search Engines
ElasticSearch, Lucene, SOLR
Monitoring
Browse, ControlBus, JMX, Log, Nagios, SNMP
OSGI
Chat
IRC, XMPP
Bus
JBI, NMR, Vert.x
Caching
cache, hazelcast
Business Process
drools, jbpmn, activiti
JAVA Message Service
ActiveMQ, JMS, SJMS, RabbitMQ, Stomp, ZeroMQ
Remote Services
RMI, Avro, Jclouds, JCIFS, SSH, EJB
Web & REST Services
CXF, CXF Bean, Restlet, Spring Web Services
Messaging
AMQP, MQTT, JT/400, Kestrel, SIP, SIPS, SMPP, SMPPS, Esper, JavaSpace
Special
geocoded,weather code
Social
Twitter, FaceBook, Yammer, DropBox
Platform support
APNS, Salesforce, SAP

Fire / Forget pattern

InOnly

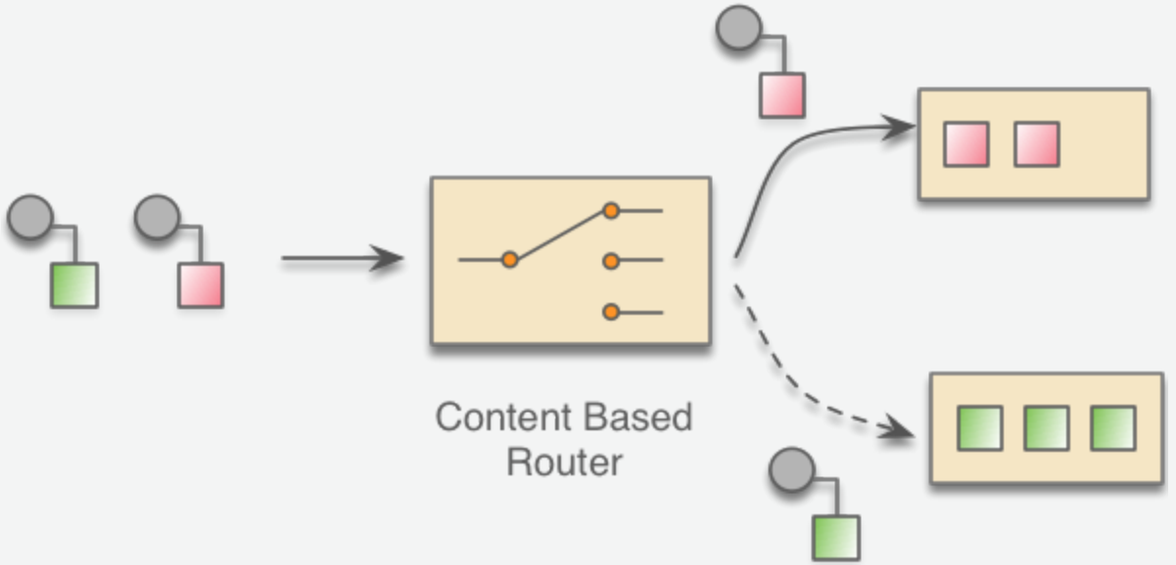


Request / Reply pattern



How To

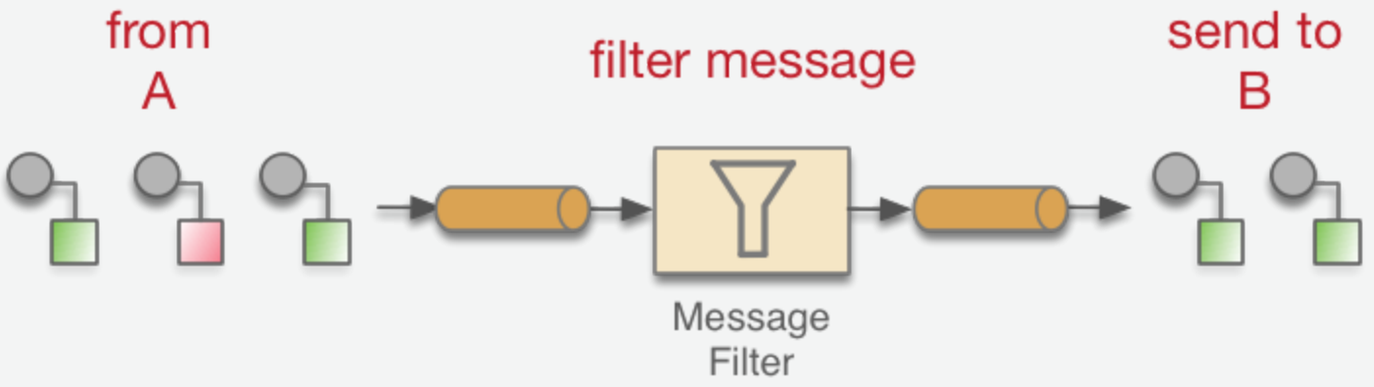
Pattern



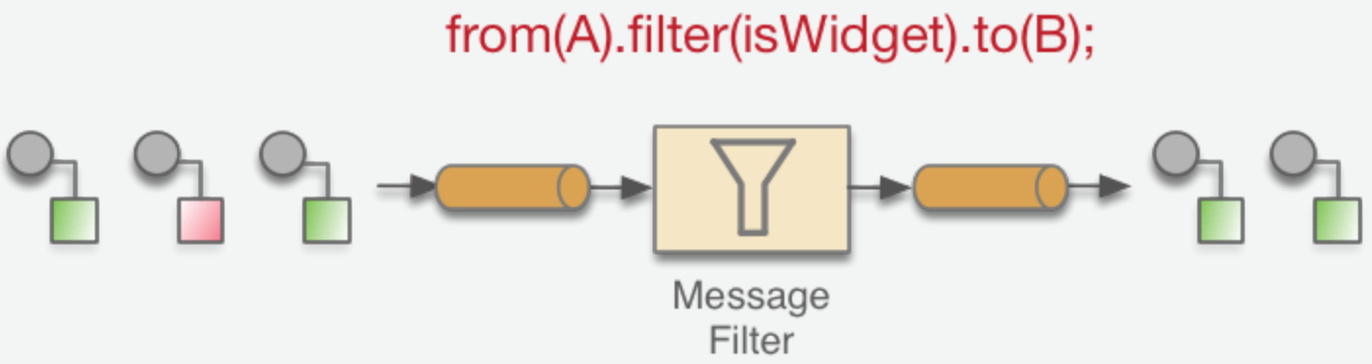
Result

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

How To

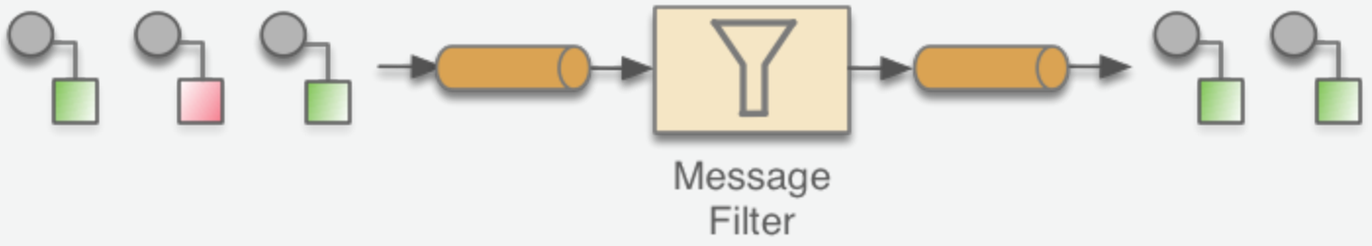


How To



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);
  }
}
```

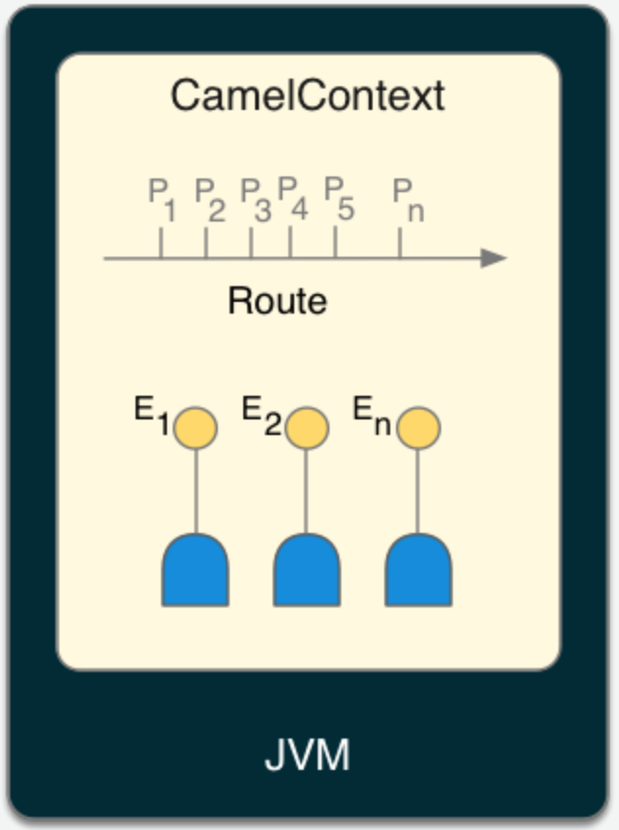
XML DSL

- 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/spri
       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>
```

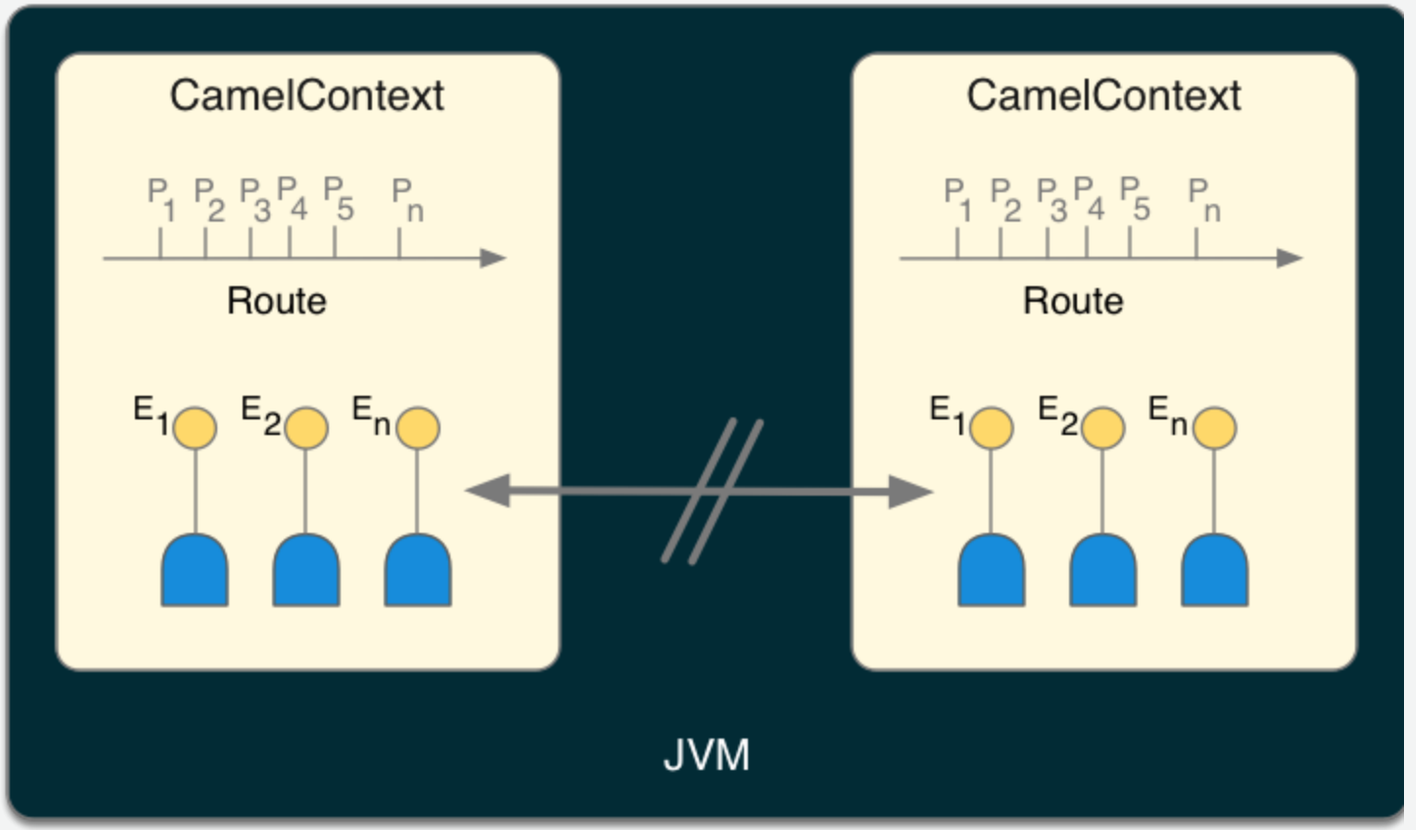
Container

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

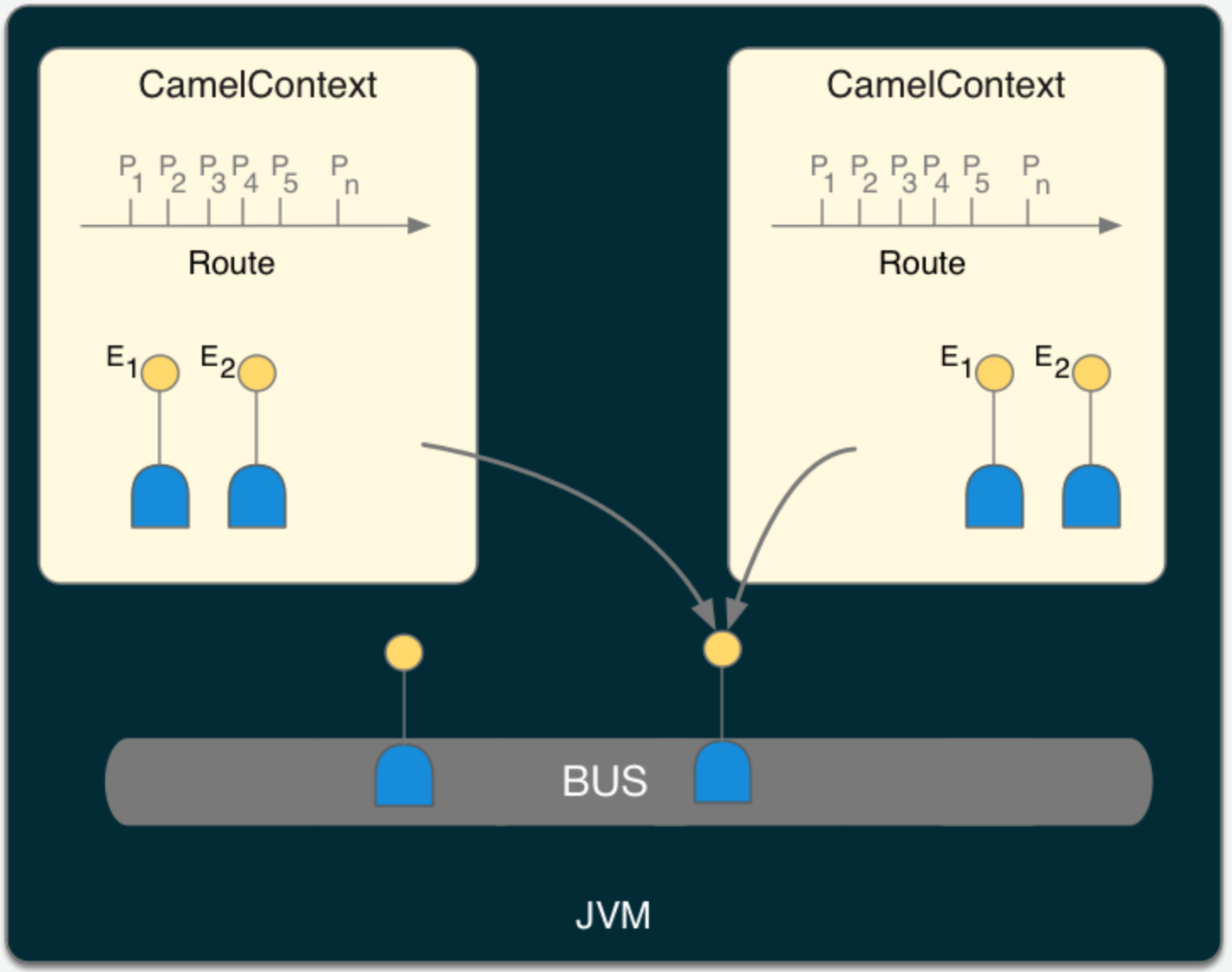


Communication

- Cross communication not allowed between context



Communication



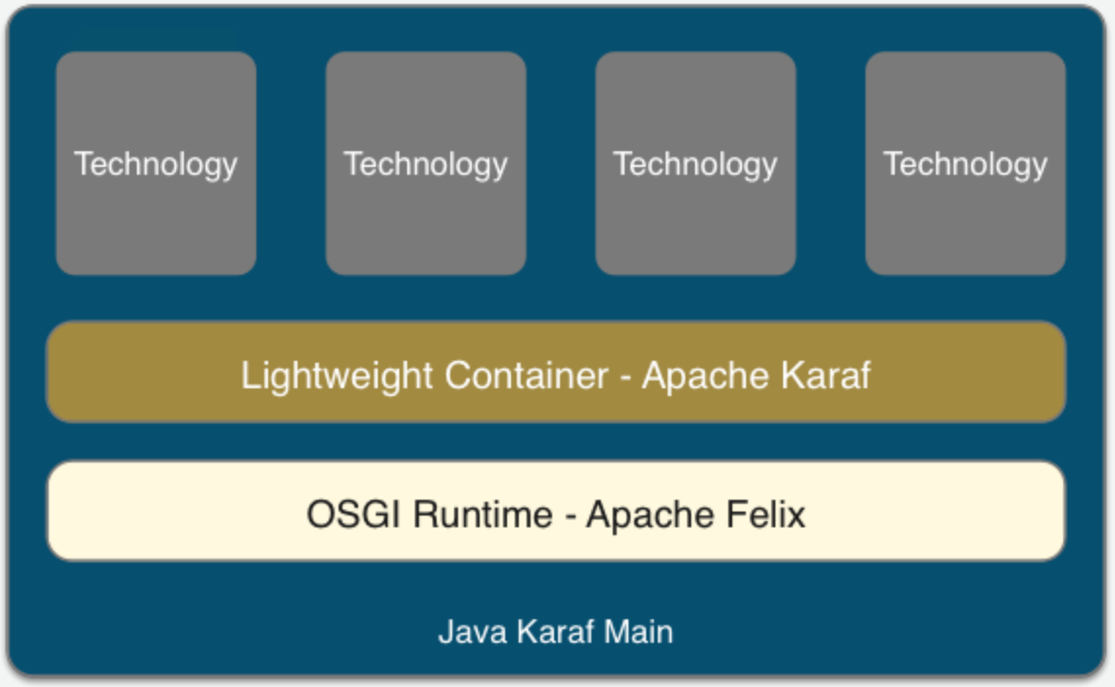
Facts

- **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

Architecture



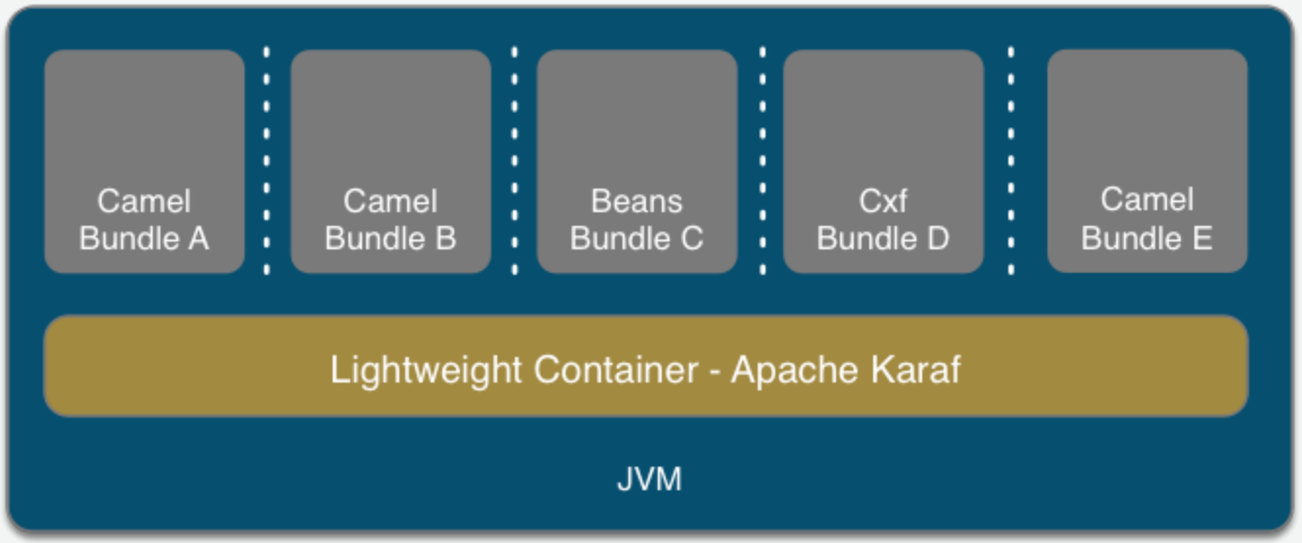
- Technology ➔ Camel, CXF, ActiveMQ, Spring, Fabric, …
- Modular platform ➔ deploy or remove container/libraries

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)

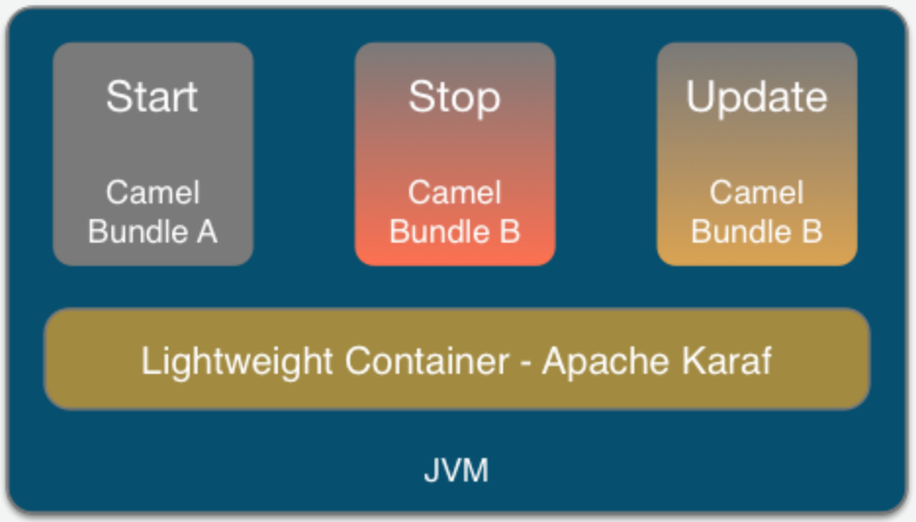
Integration container

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



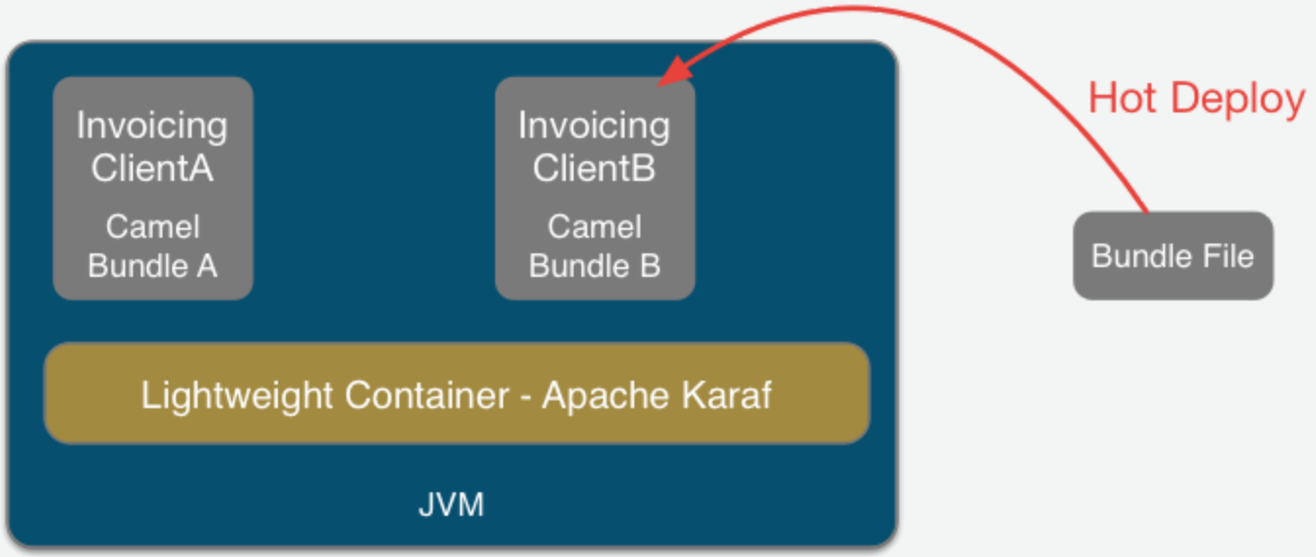
Integration container

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

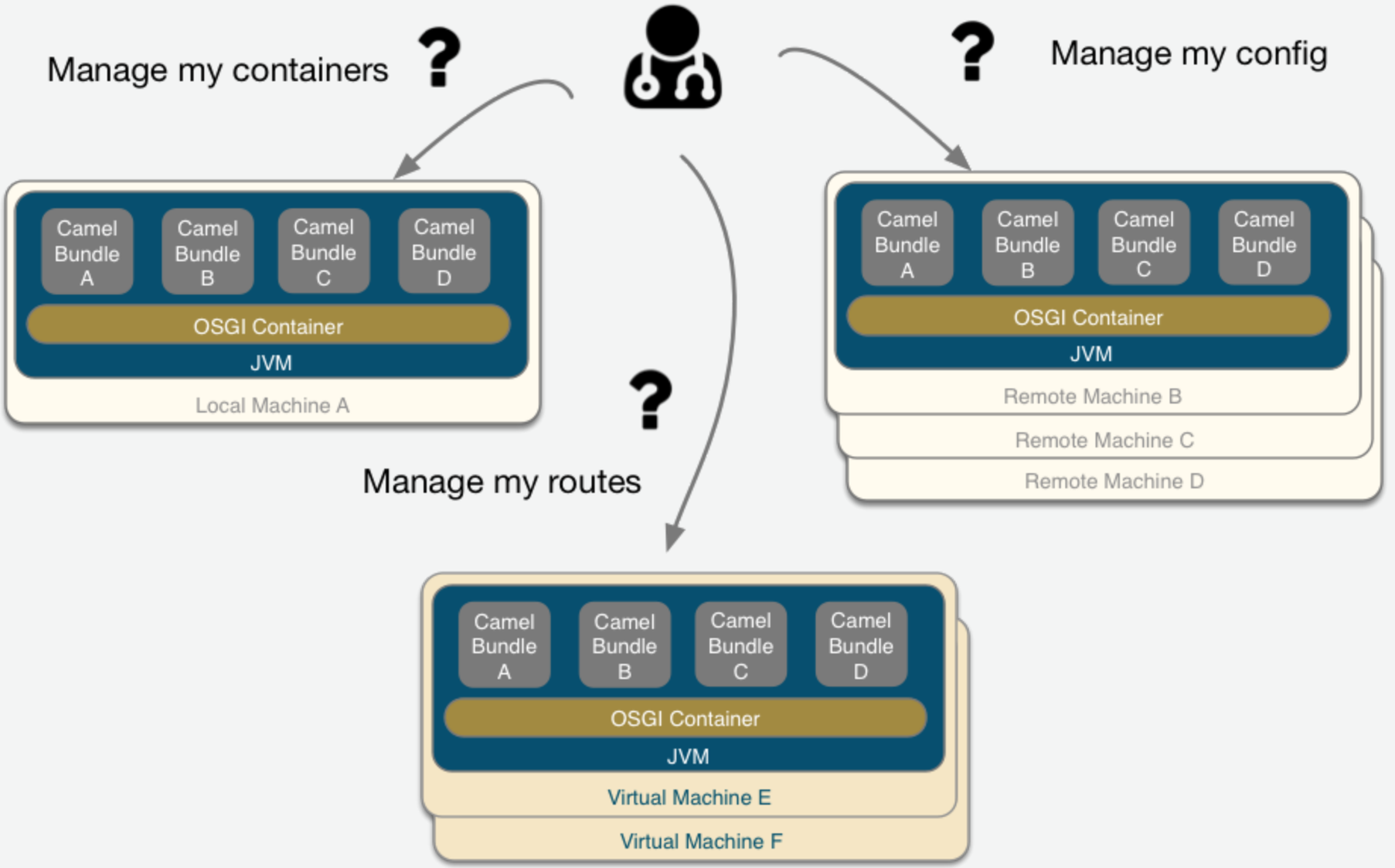


Integration container

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

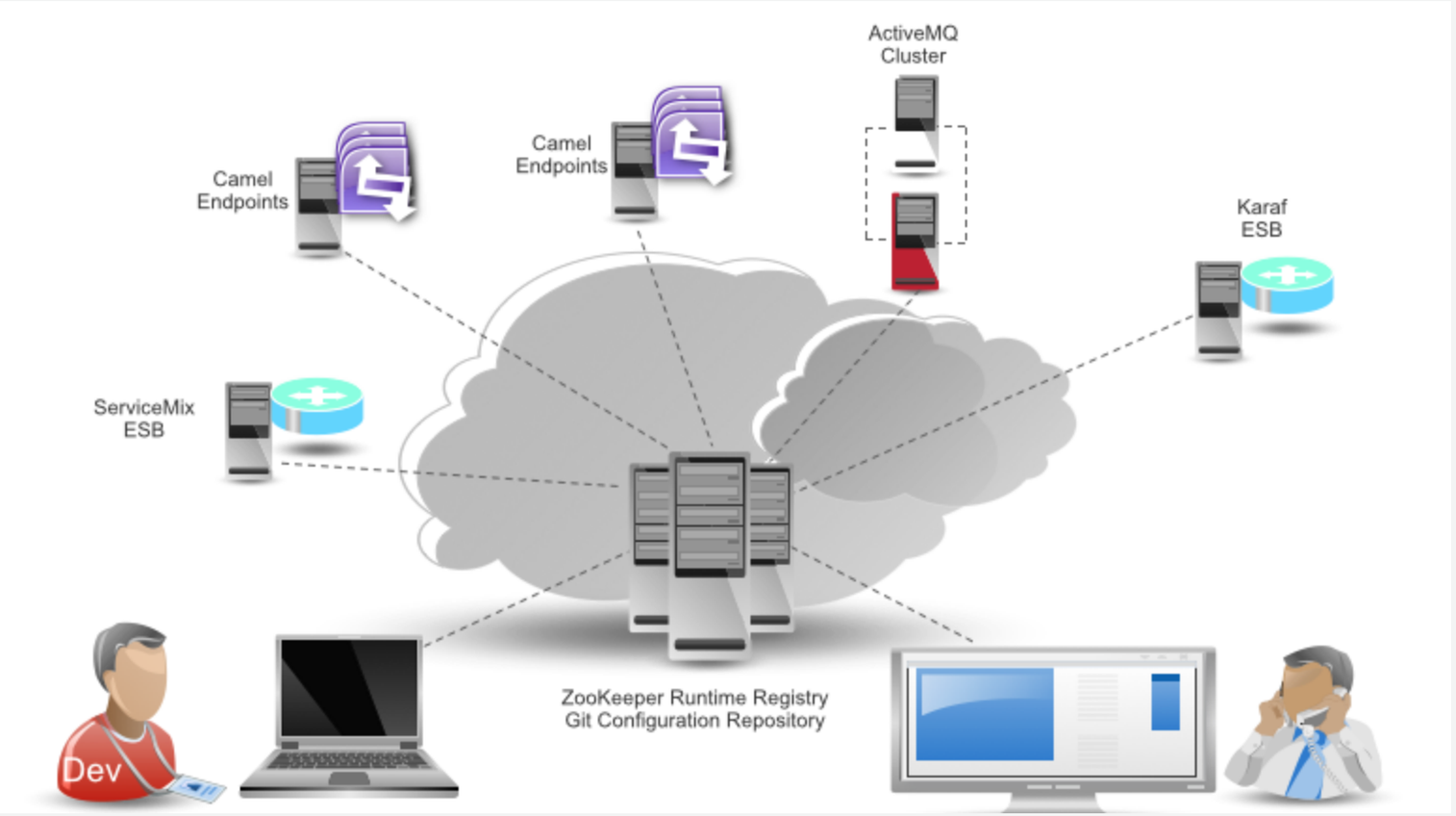


(Cloud) Deployment



Fabric8 v1

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

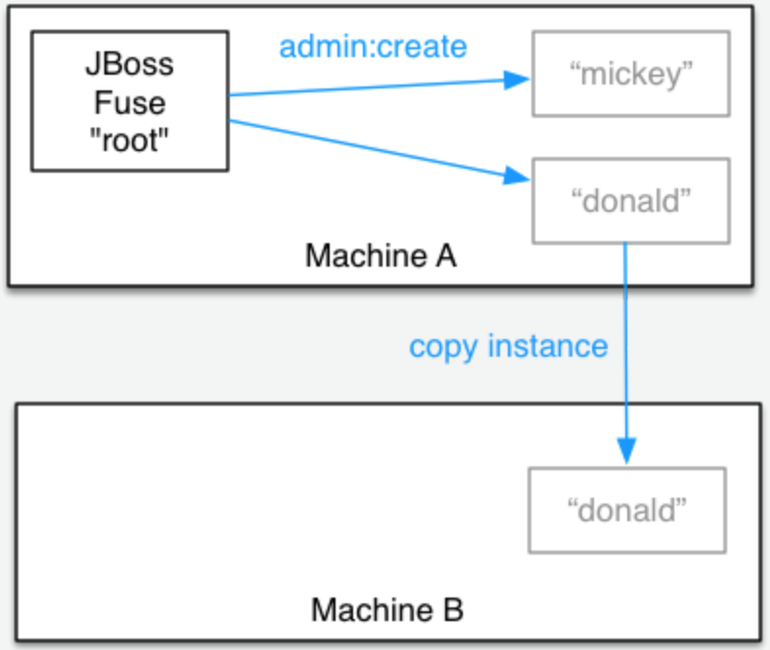


Features

- **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

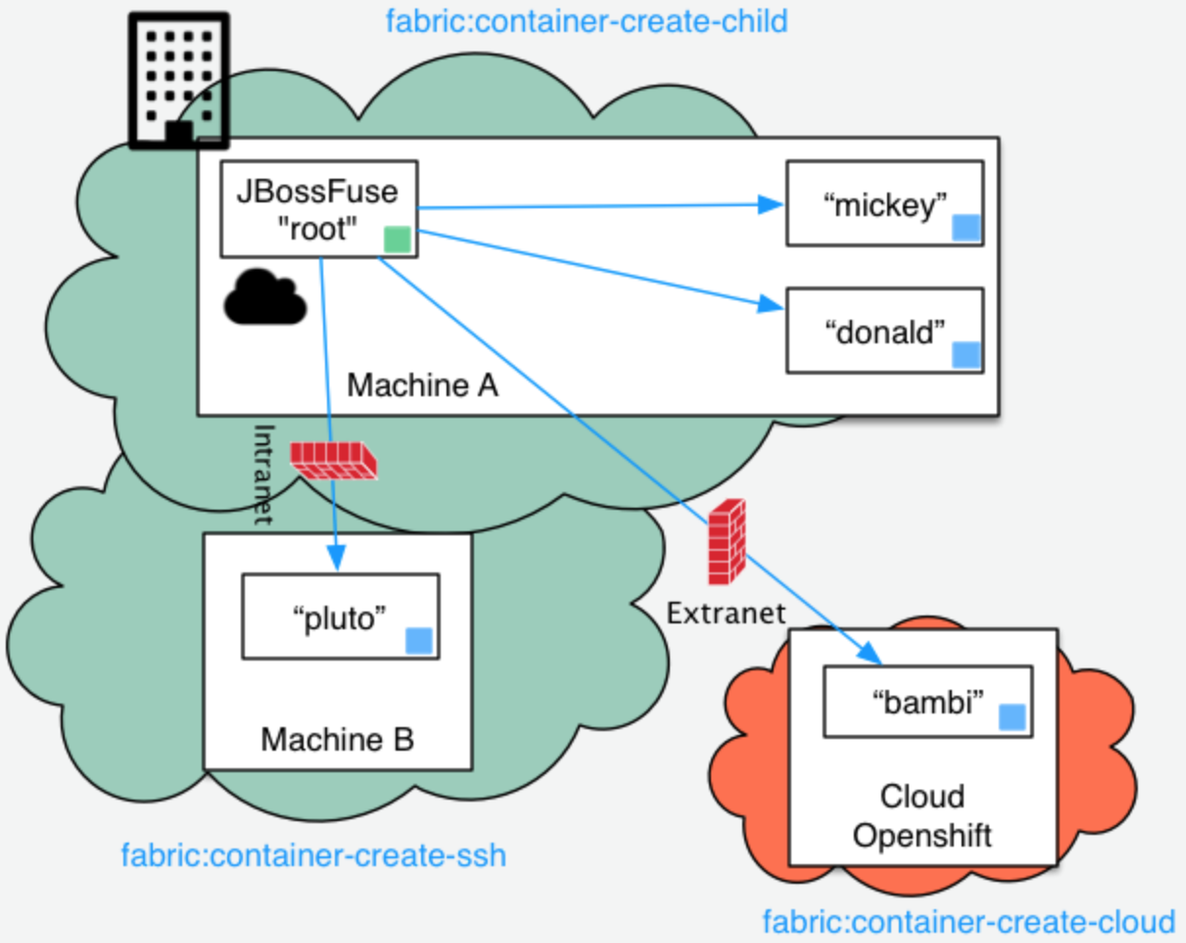
Karaf limitations

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



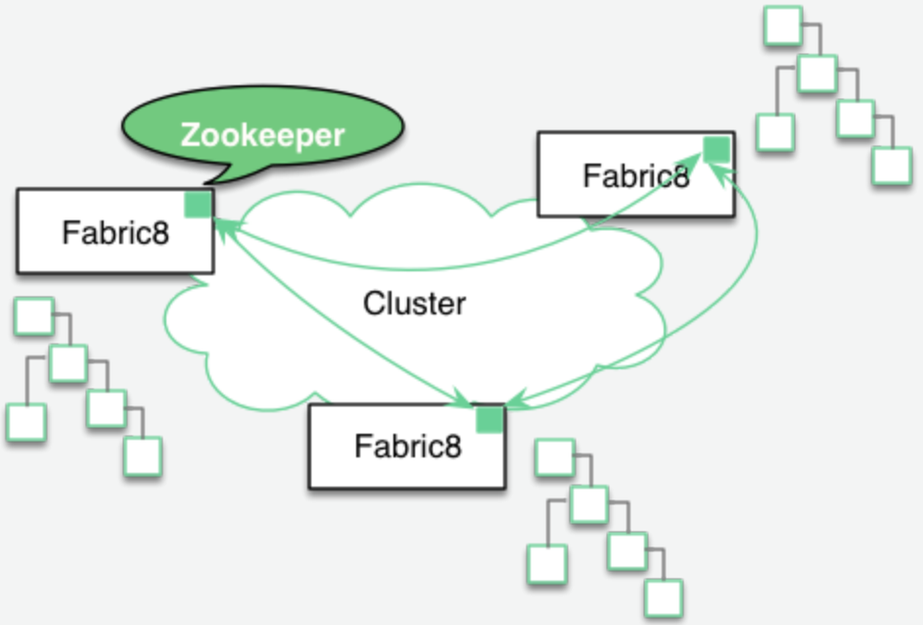
- Instances are **not cloned**
- **Configurations** must be managed (manually, script)

Fabric extends the possibilities



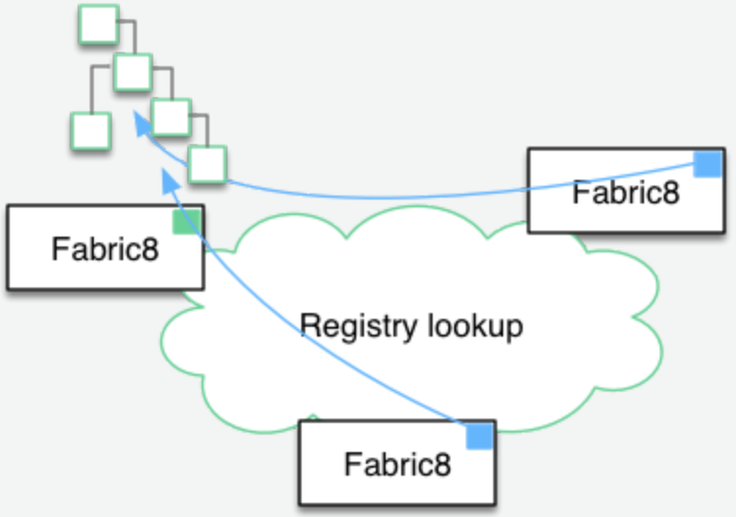
Coordinating System : Zoo

- **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

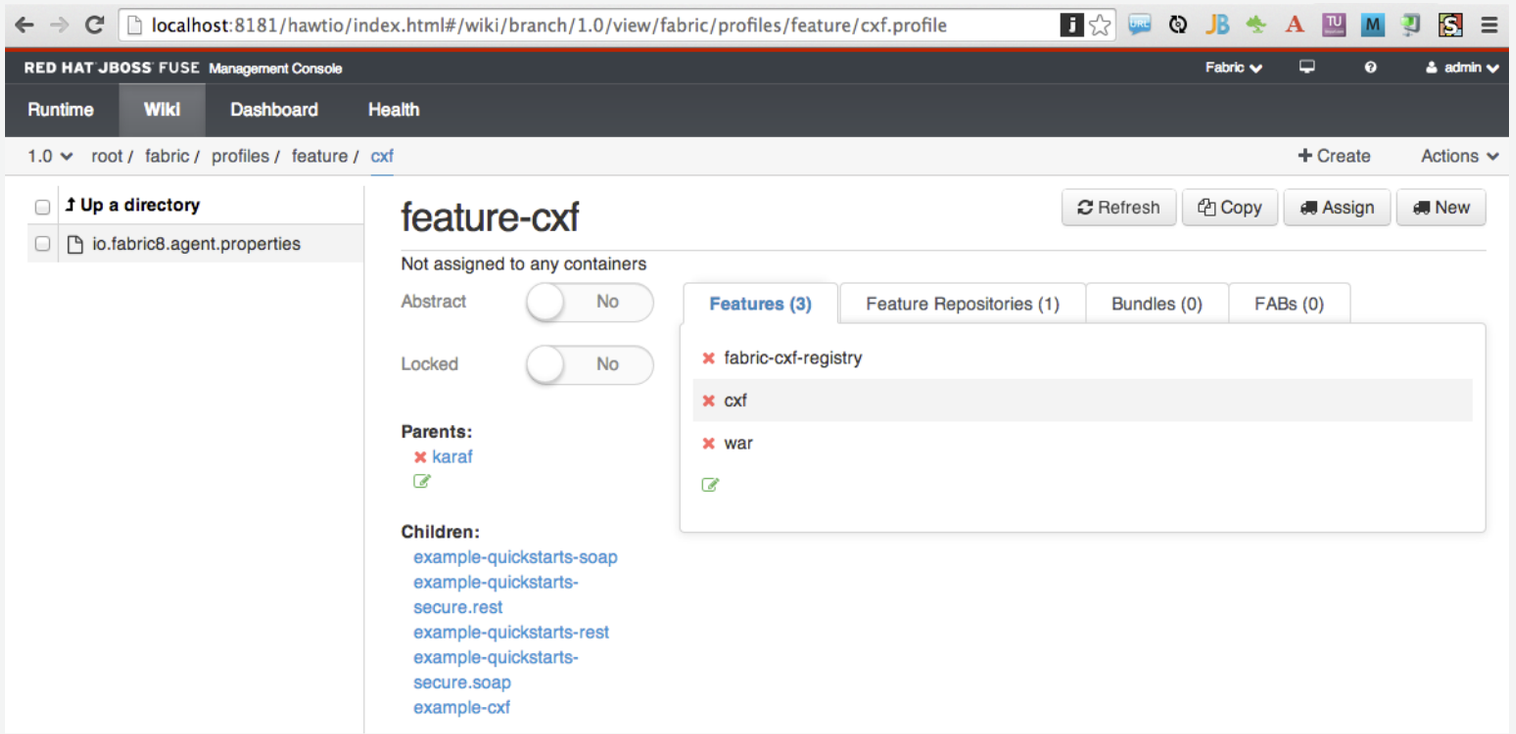
- Are the **clients** of the Zookeeper server(s)



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

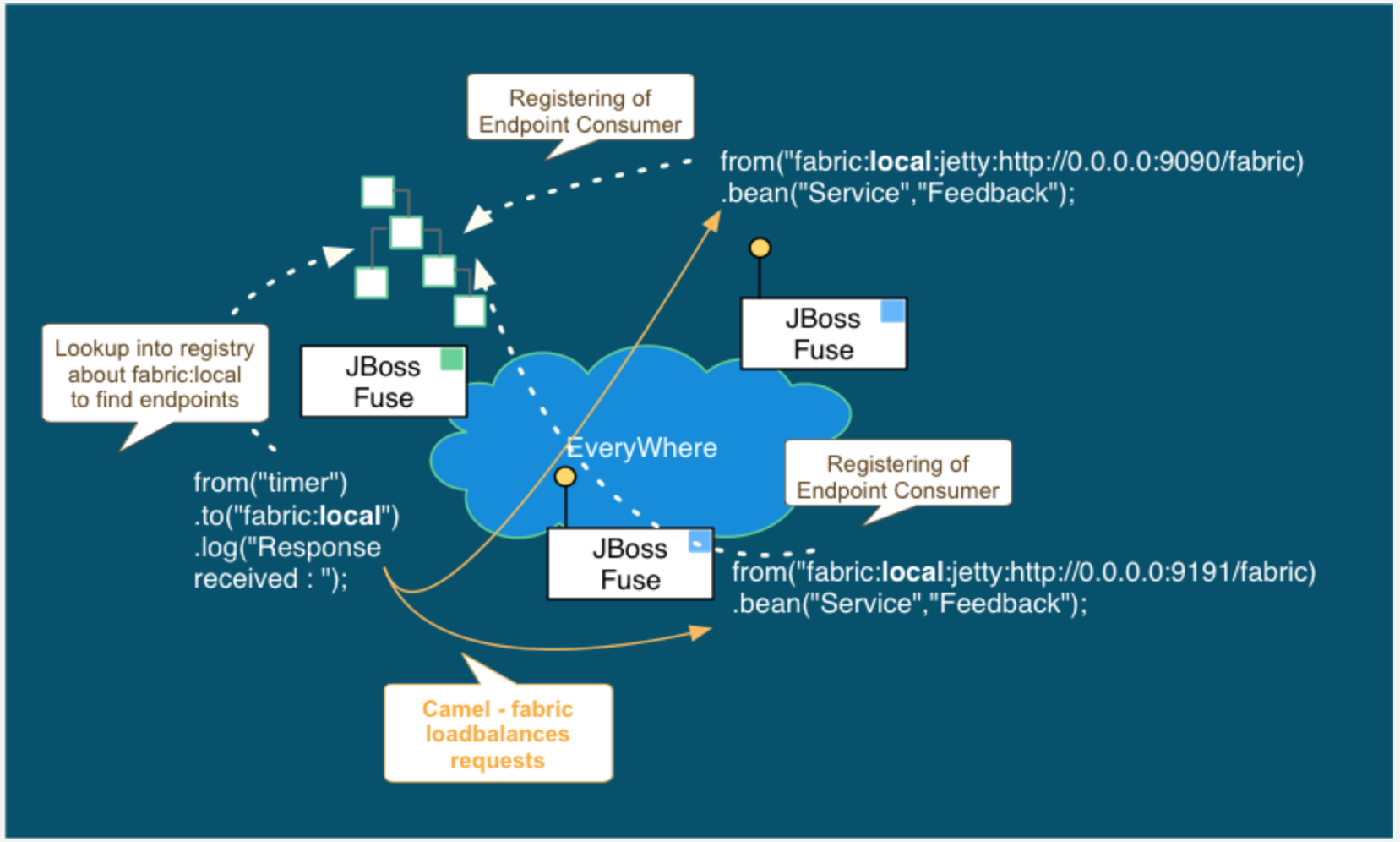
Profiles

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



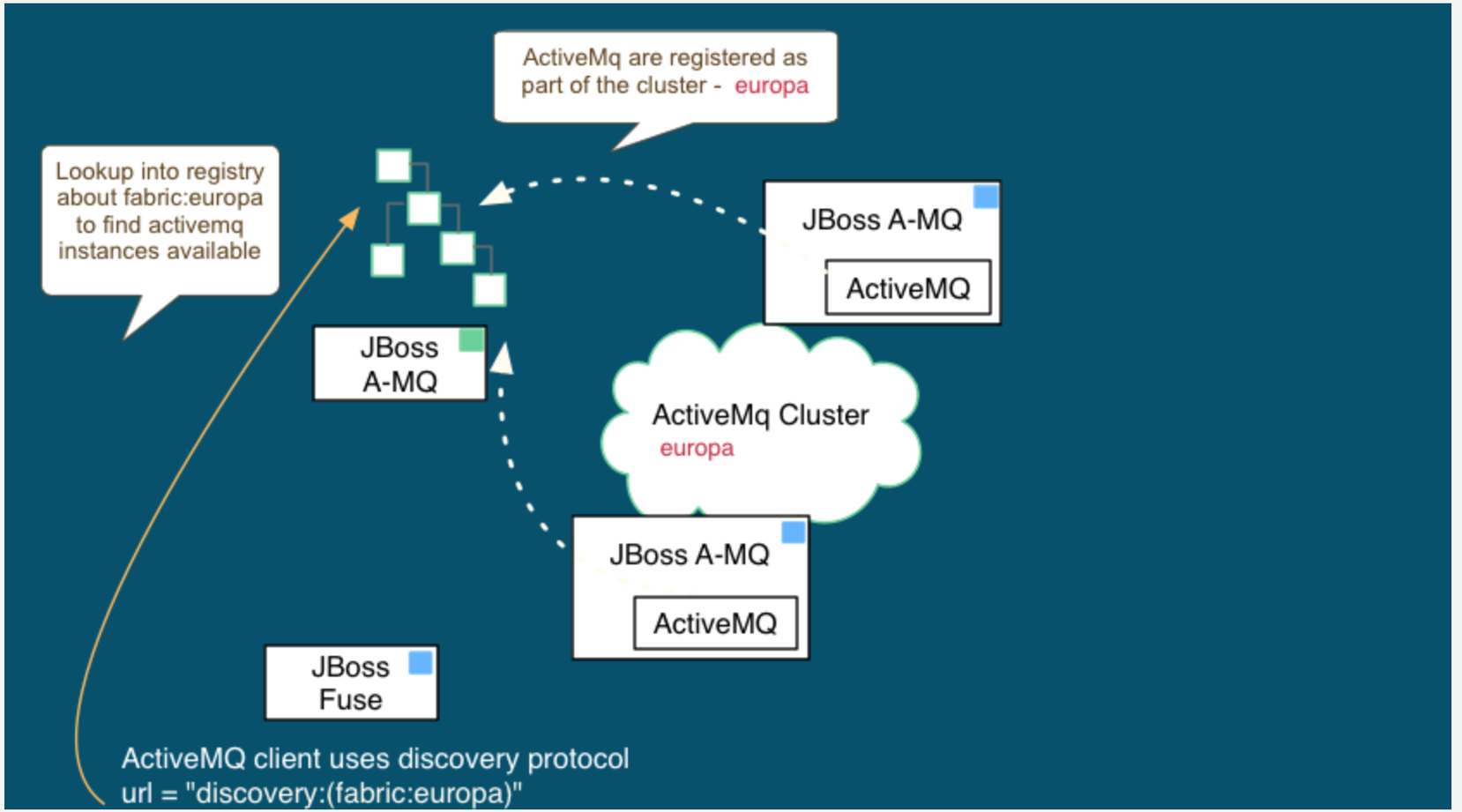
Virtualization & Load balancing

- 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

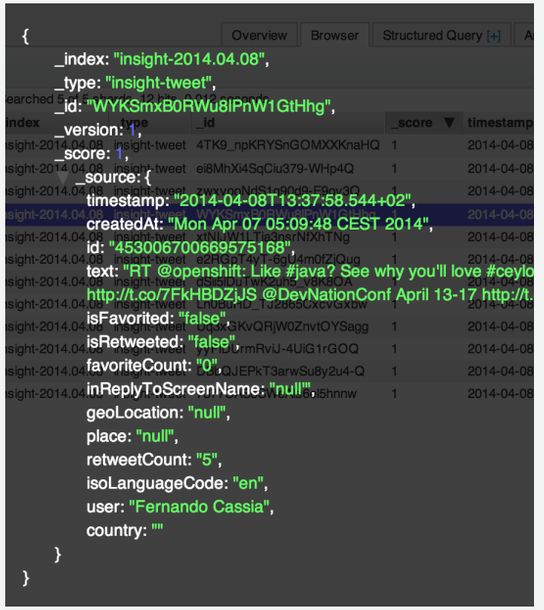
- **Full Text Search** features ➔

Lucene

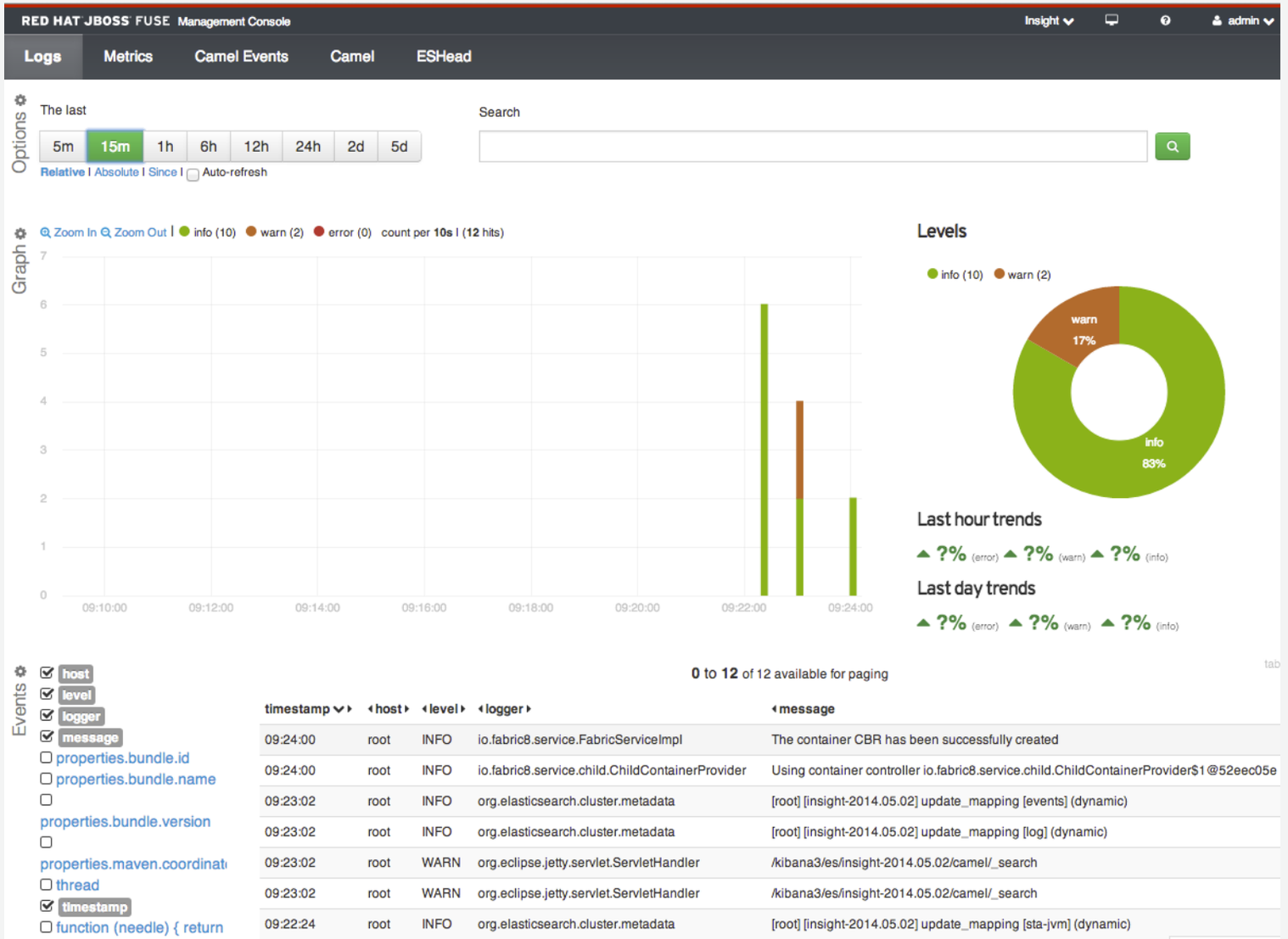
Info collected

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

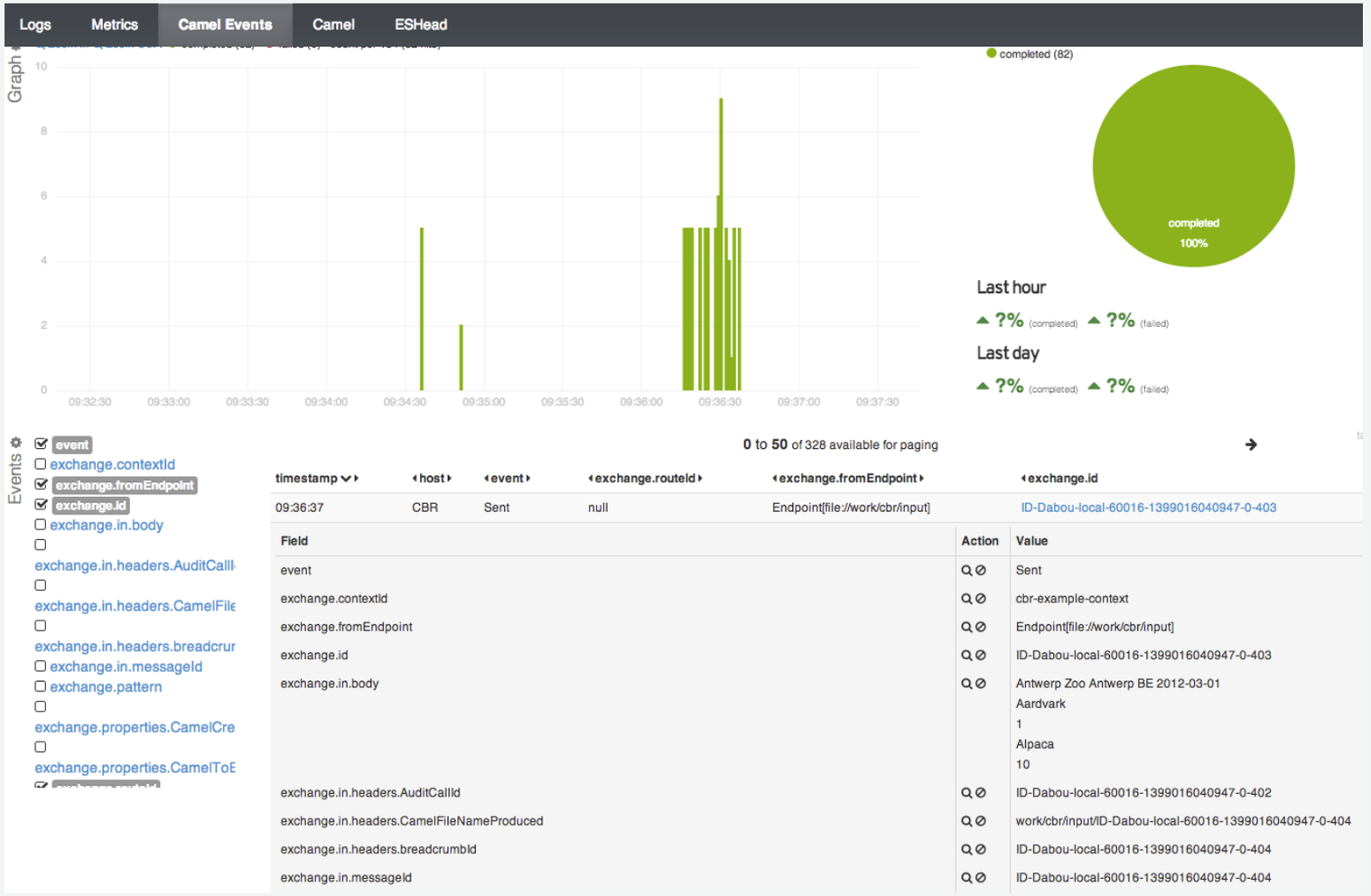
```
import org.apache.camel.Header;
import org.fusesource.insight.storage.StorageService;
import java.sql.Timestamp;
import java.util.Date;

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

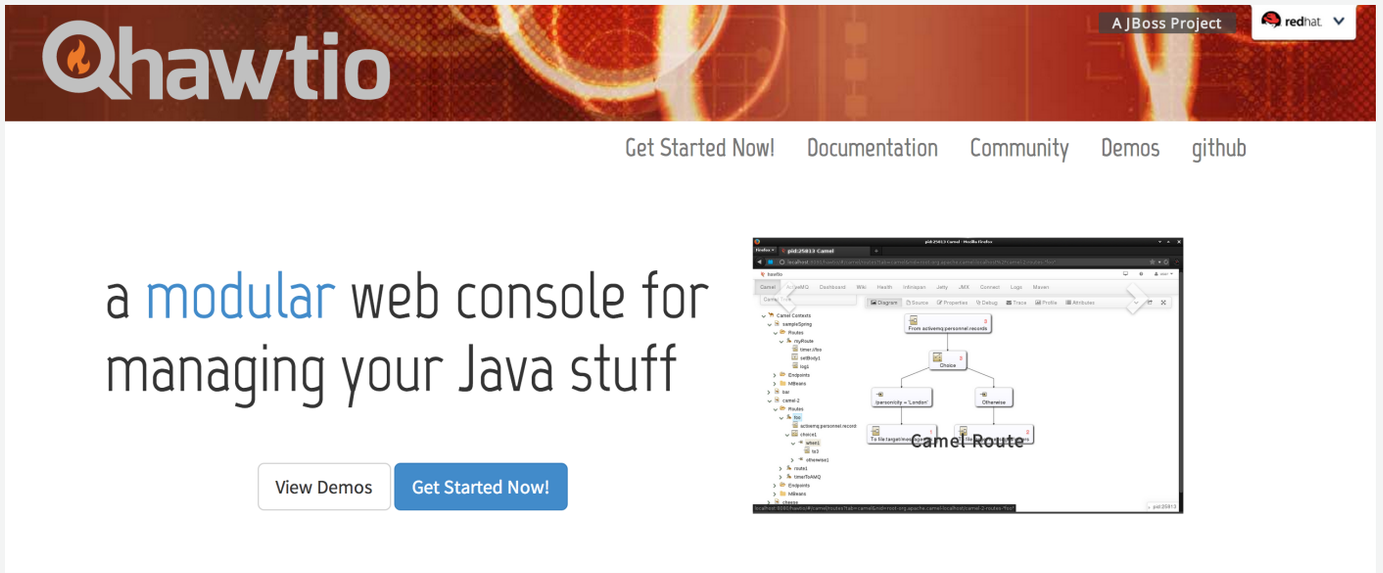


Insight Camel



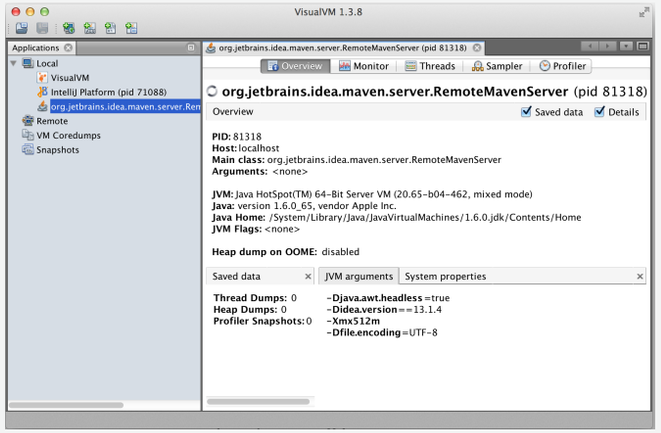
Hawt.io

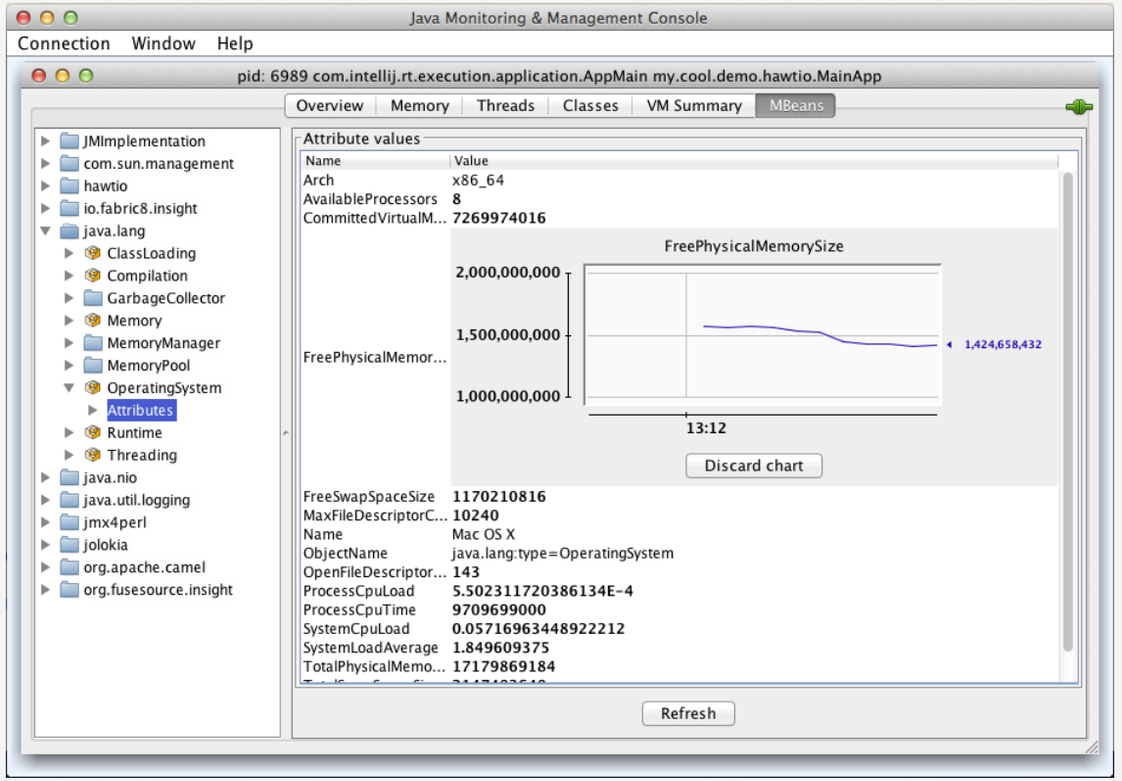
- **OpenSource** project - Apache License
- **Pluggable** & **modular web console** for managing Java **MBeans**



JMX Tools

- JConsole, VisualVM ➔ Developers tool (🚑)
- Proprietary `com` protocol (RMI/IIOP)

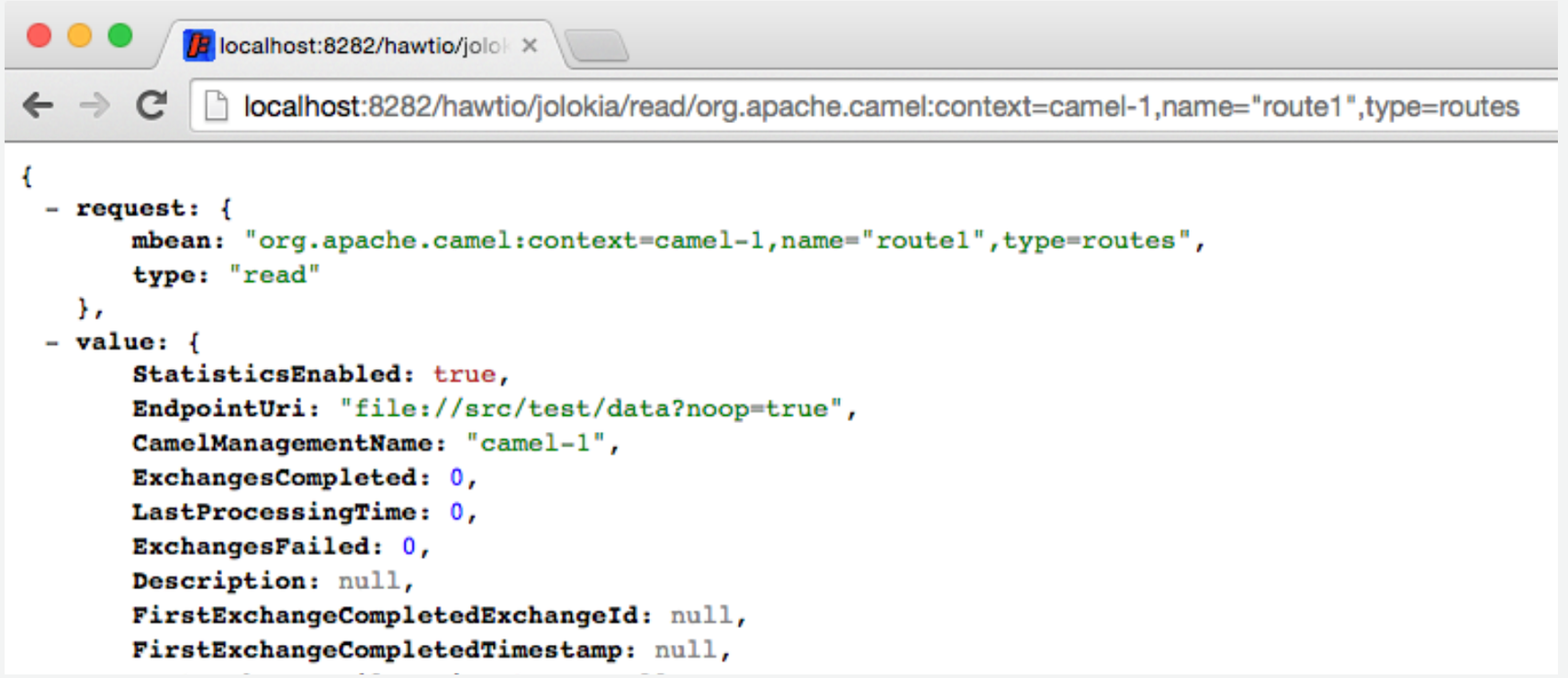




- Command line tools ➔ jmxclient, jmxterm, mx4j
- Not so user friendly : ☹

Jolokia ?

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



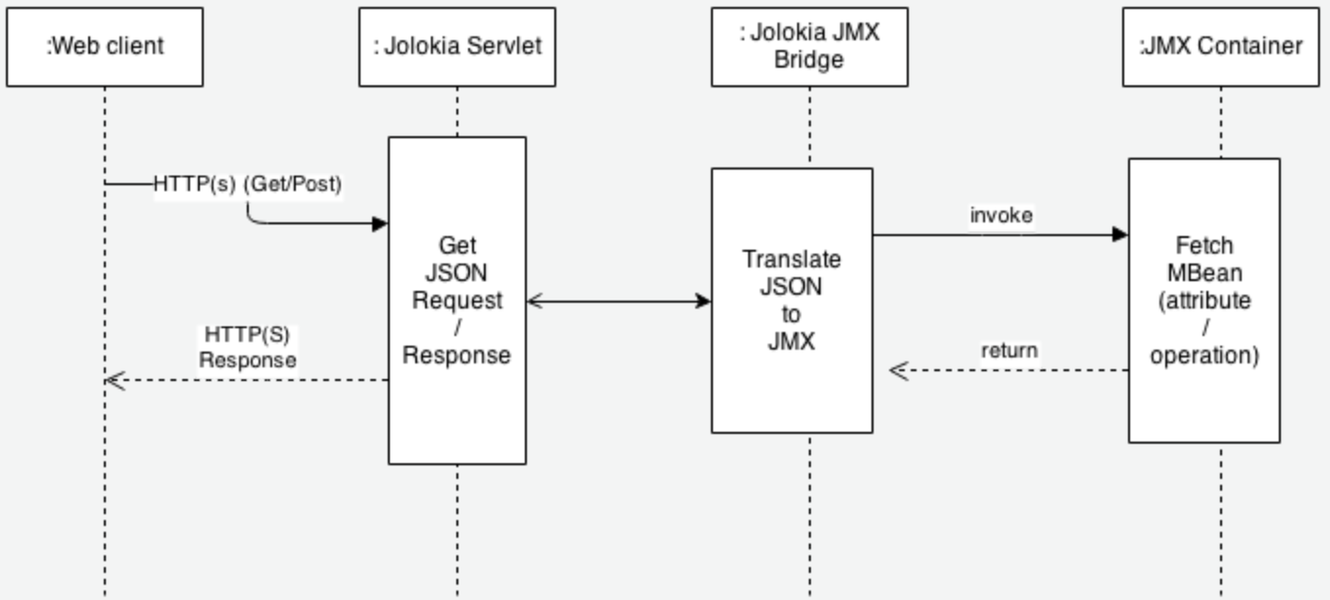
Modern HTML5 Architecture

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

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

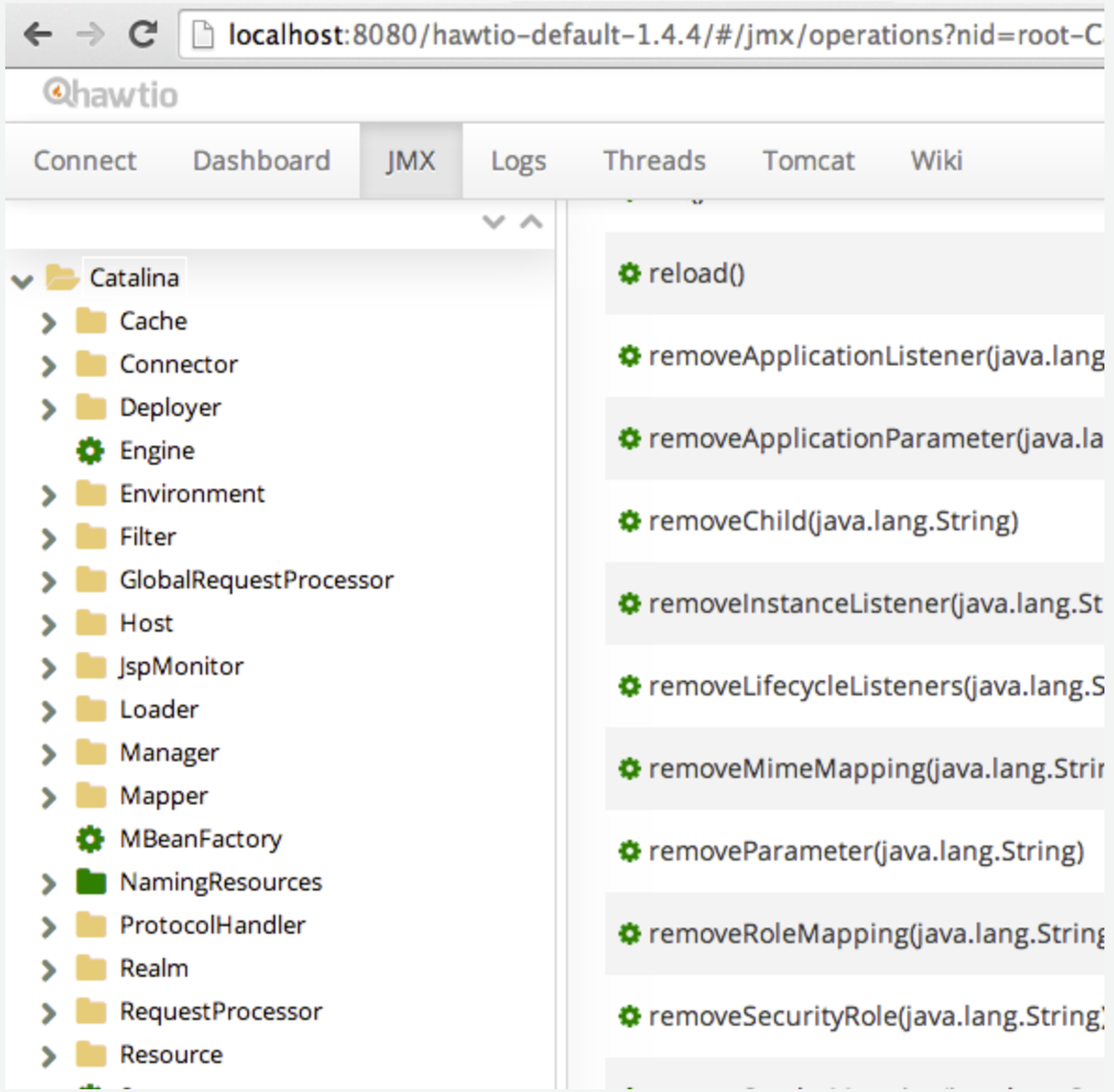
Communication …

- **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



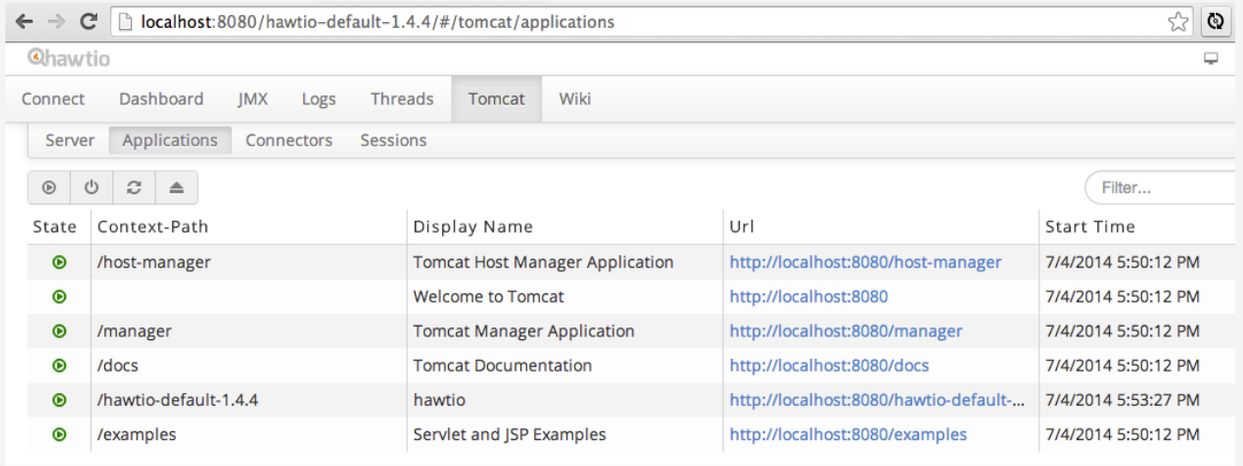
Domain

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



Plugin

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



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="Star
                      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="Ref
                      class="icon-refresh"></i></button>
              <button ng-disabled="selected.length == 0" class="btn" ng-click="uninstallDialog.oper
                      class="icon-eject"></i></button>
            </div>
          </div>
        </fieldset>
      </form>
```

(1) Angular ng-controller defined within HTML 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'}).
      when('/tomcat/connectors', {templateUrl: 'app/tomcat/html/connectors.html'}).
      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,*");
  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));
}
```

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'];
```

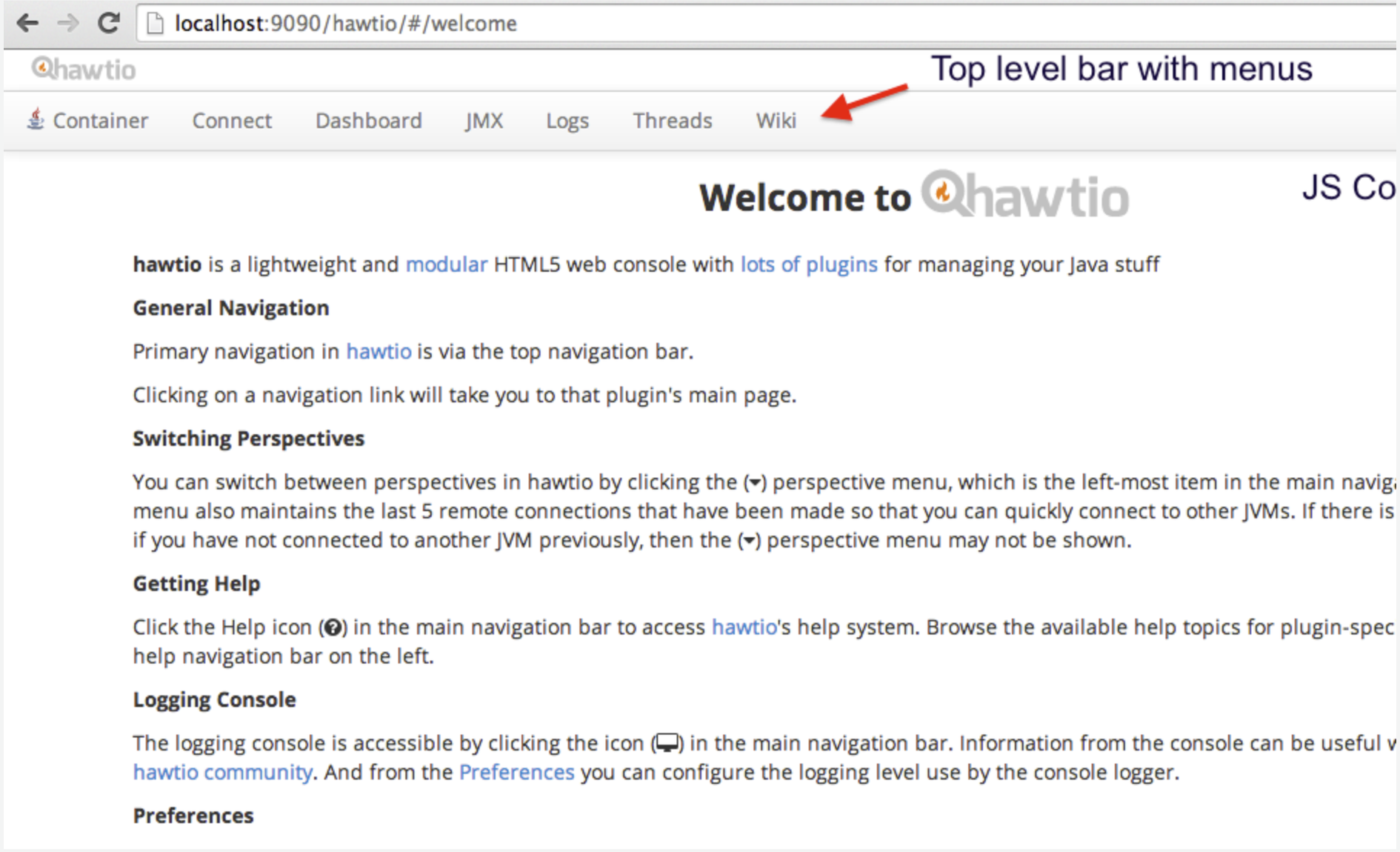
1. Response rendered & parsed

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

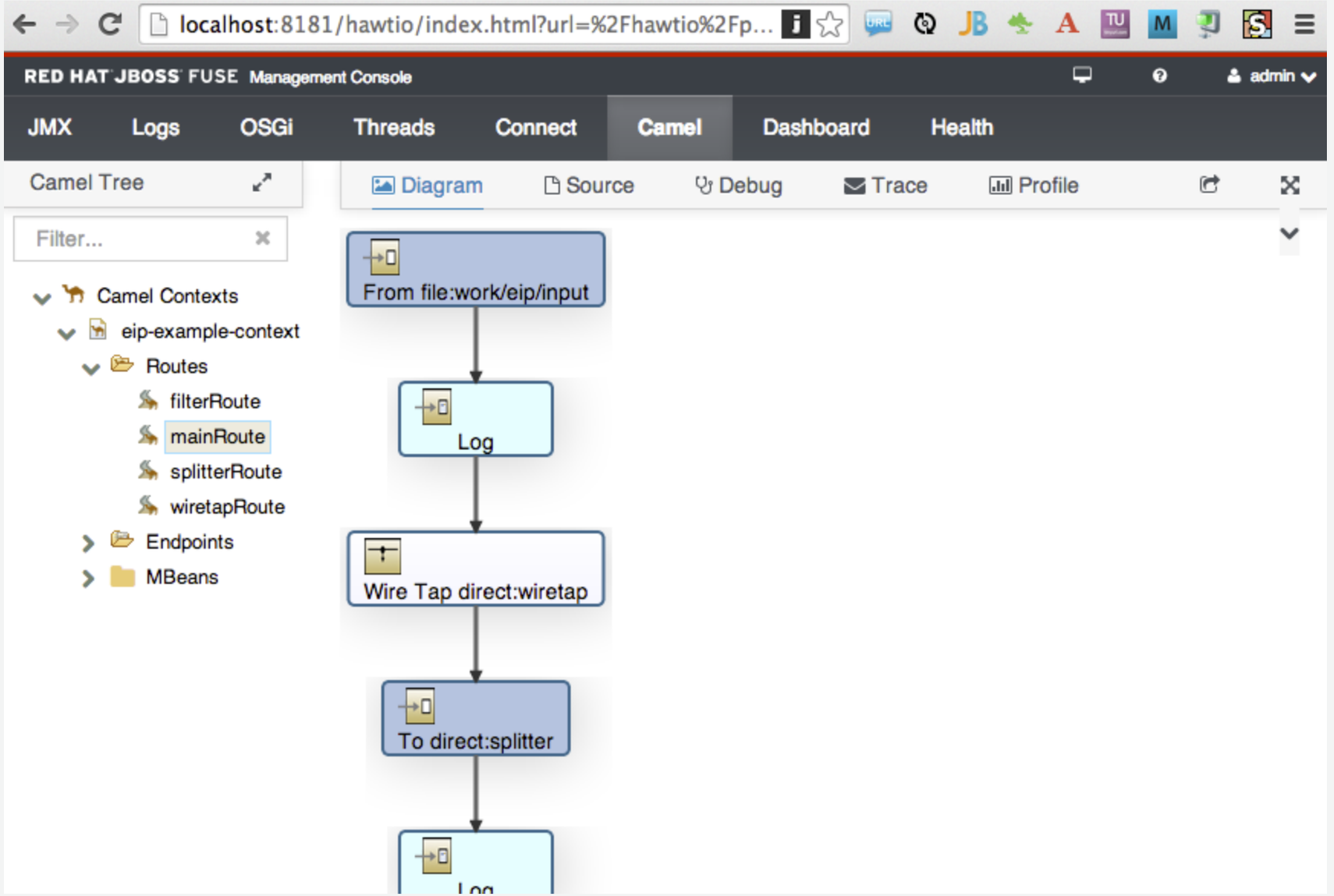
Plugins

- **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)

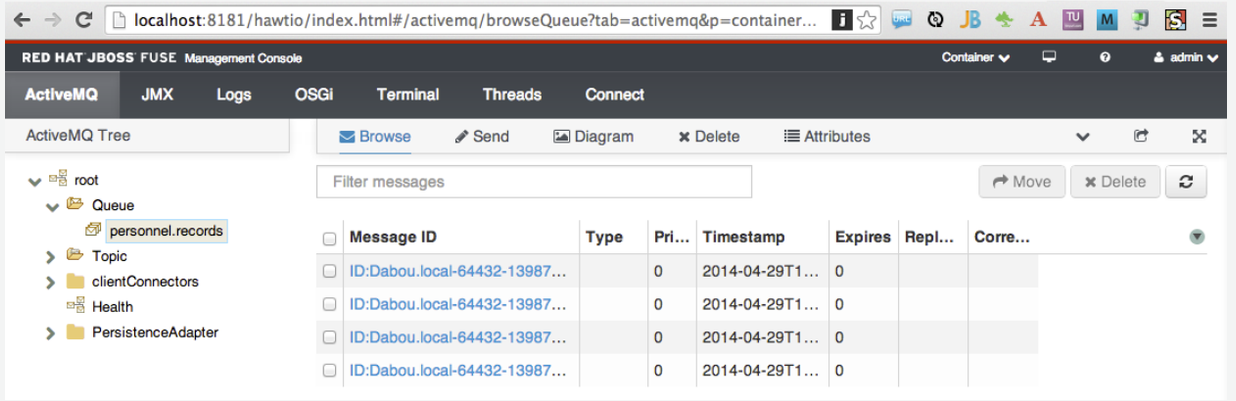
What Front looks like

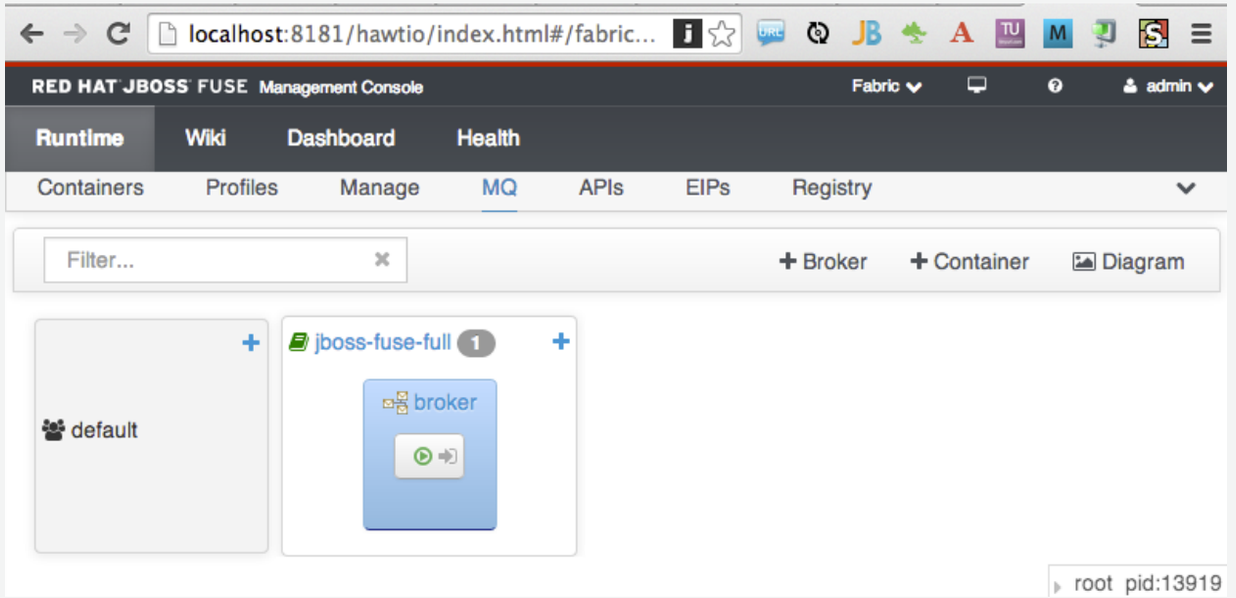


Discover your camel routes

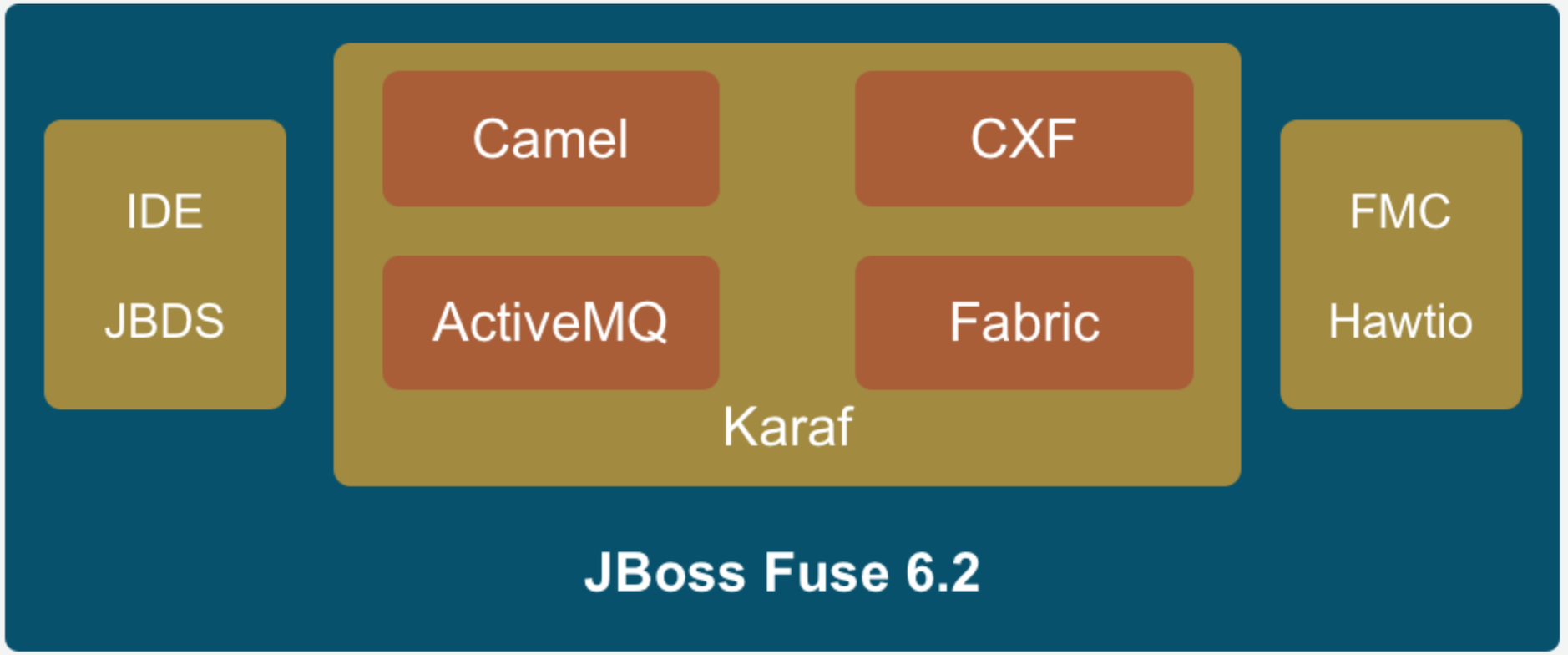


Manage your brokers



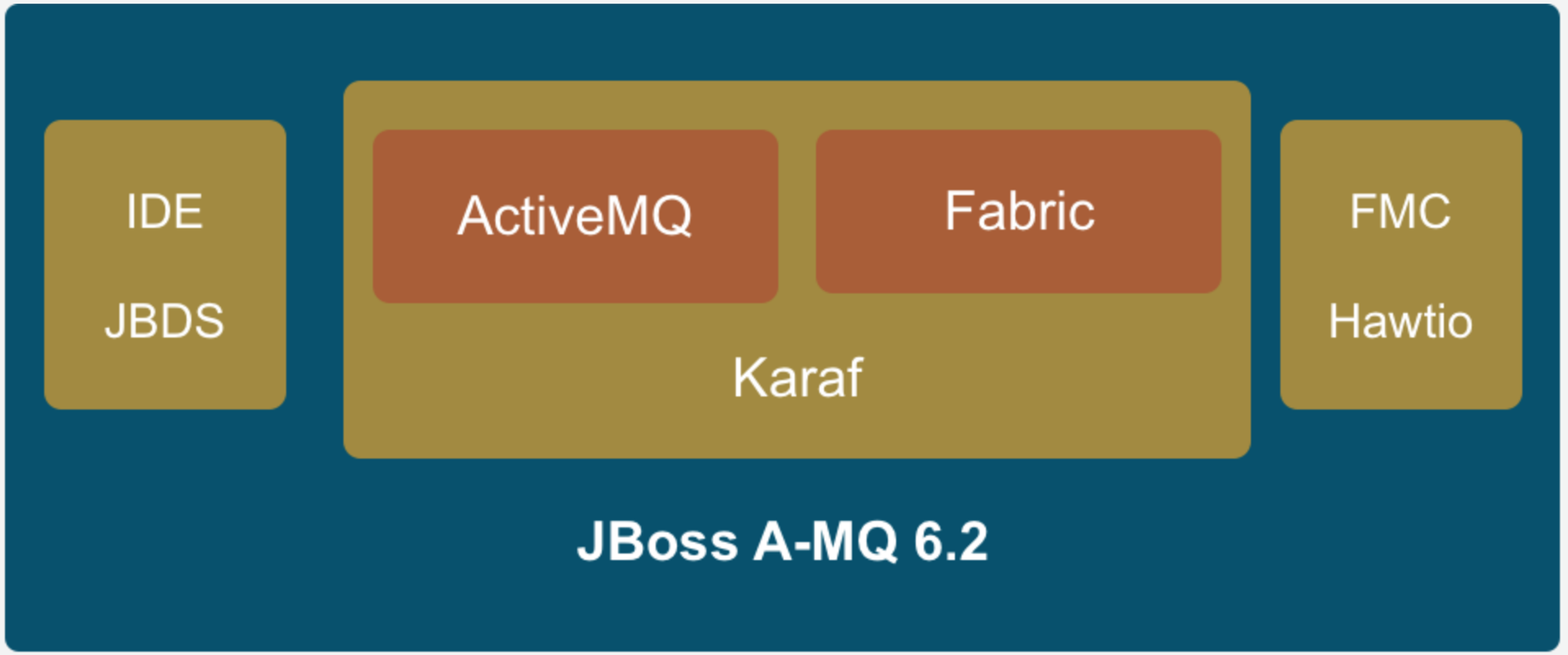


Integration & Mediation Platform



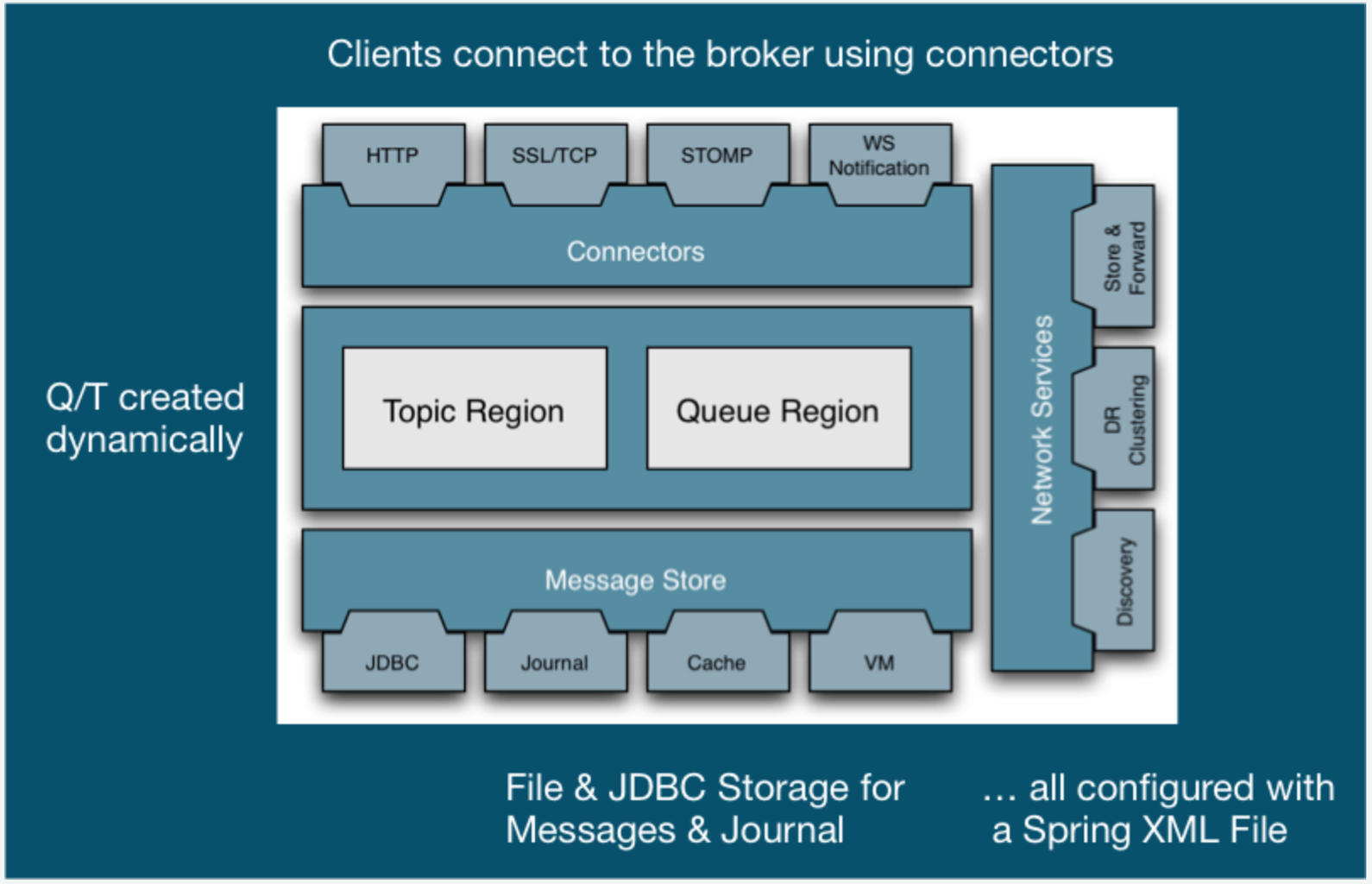
- **> 100** components : file, jms, ftp, …
- **> 50** EIP Patterns : content based router, splitter, aggregator, …

Messaging Broker Platform



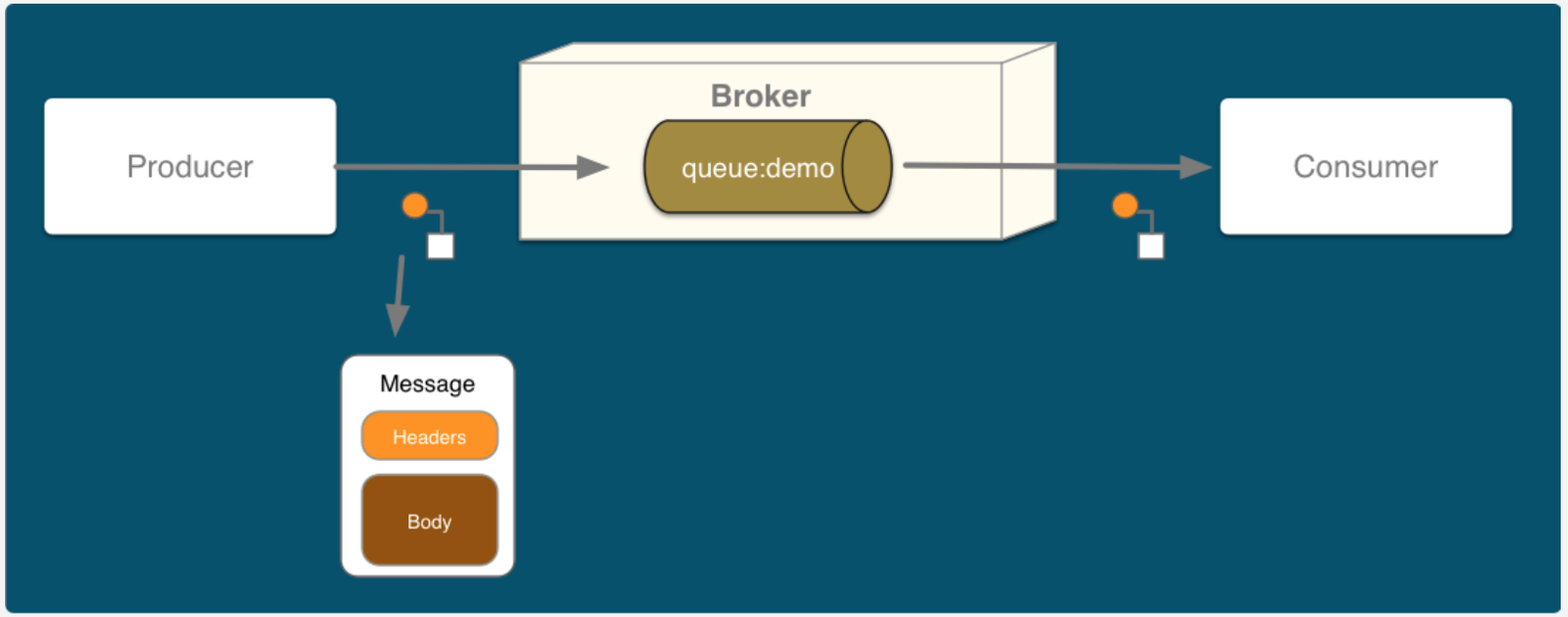
- Support : JMS, AMQP, MQTT, STOMP, Websocket
- **iPaas** (Openshift cartridges - v2)

Broker Architecture



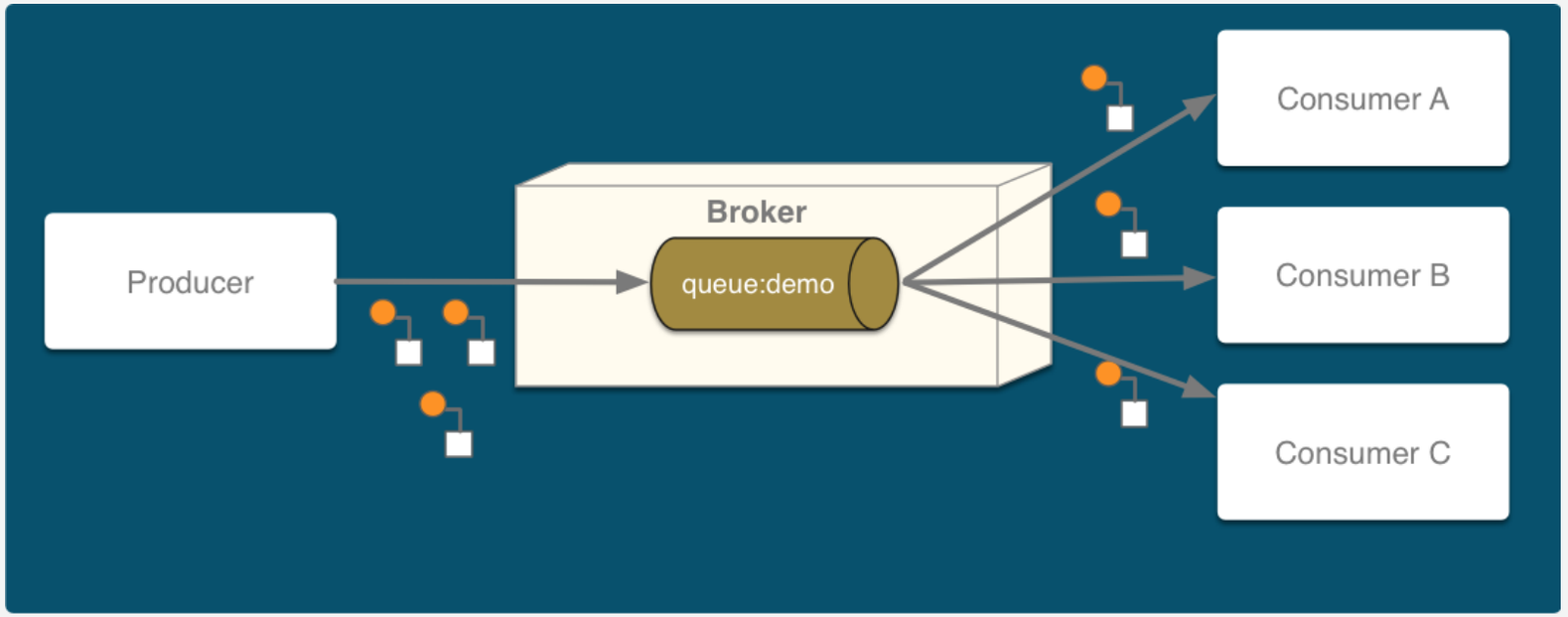
Point-to-Point Messaging

- 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



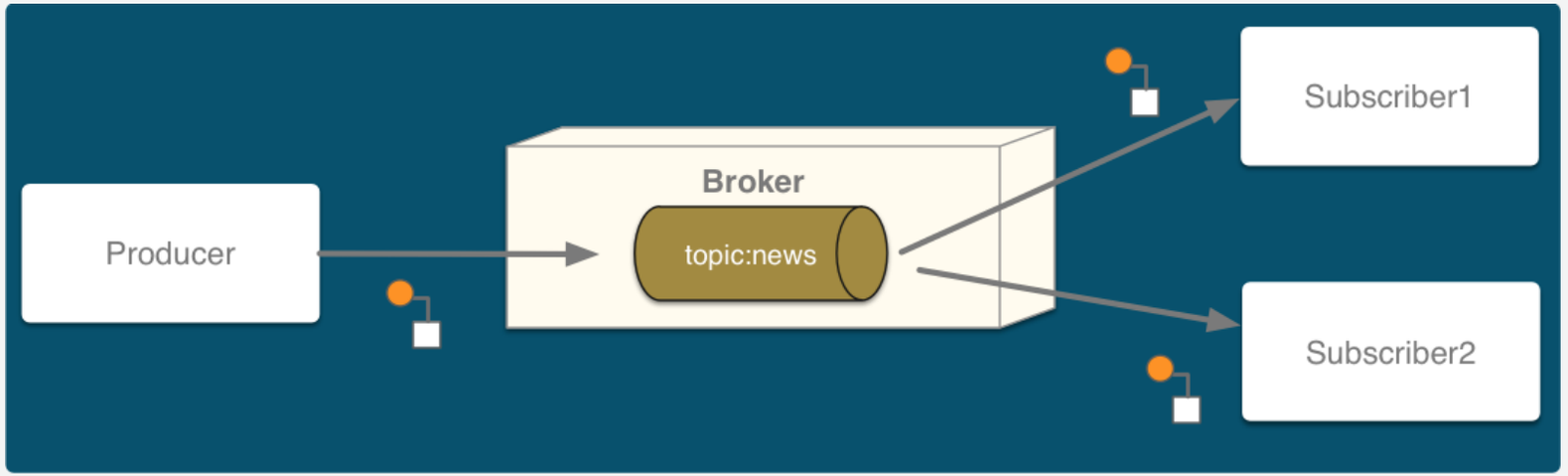
Load Balancing

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



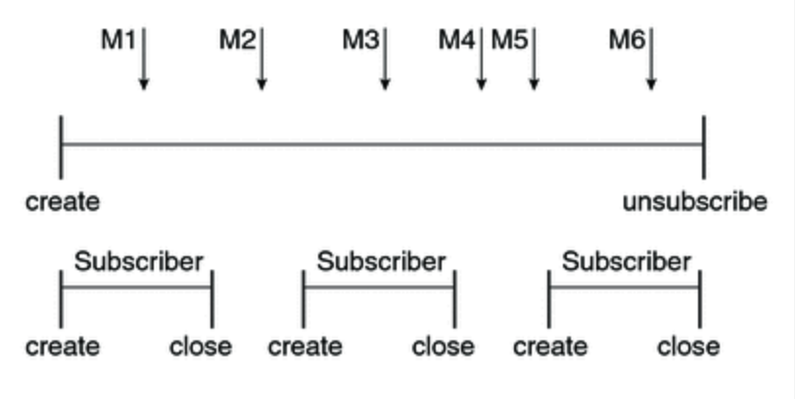
Publish and Subscribe

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



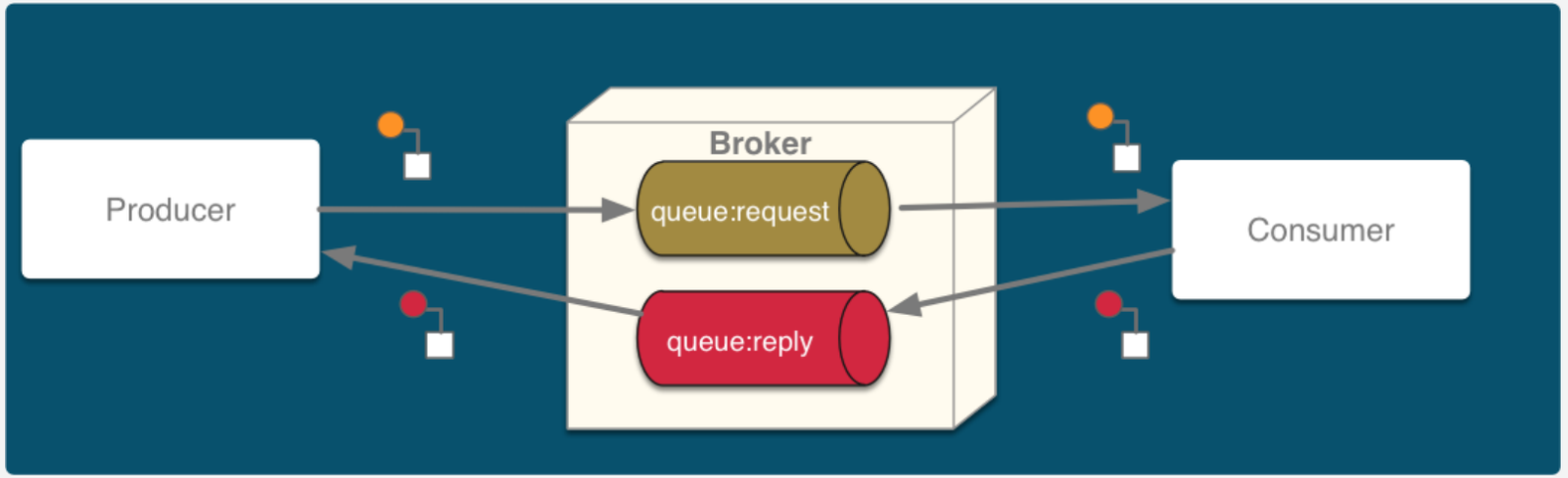
Durable Subscriber

- 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



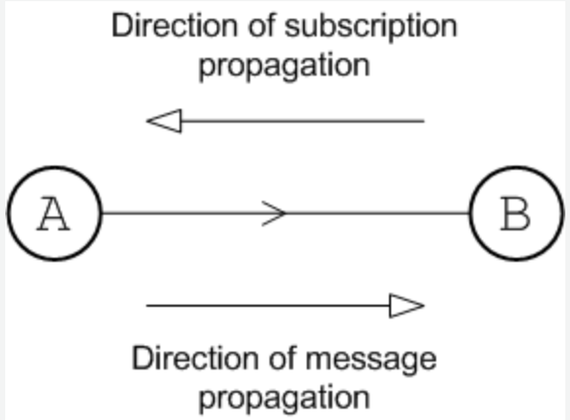
Request/Reply

- 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

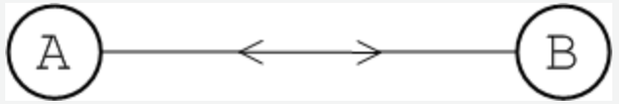


Network of Brokers

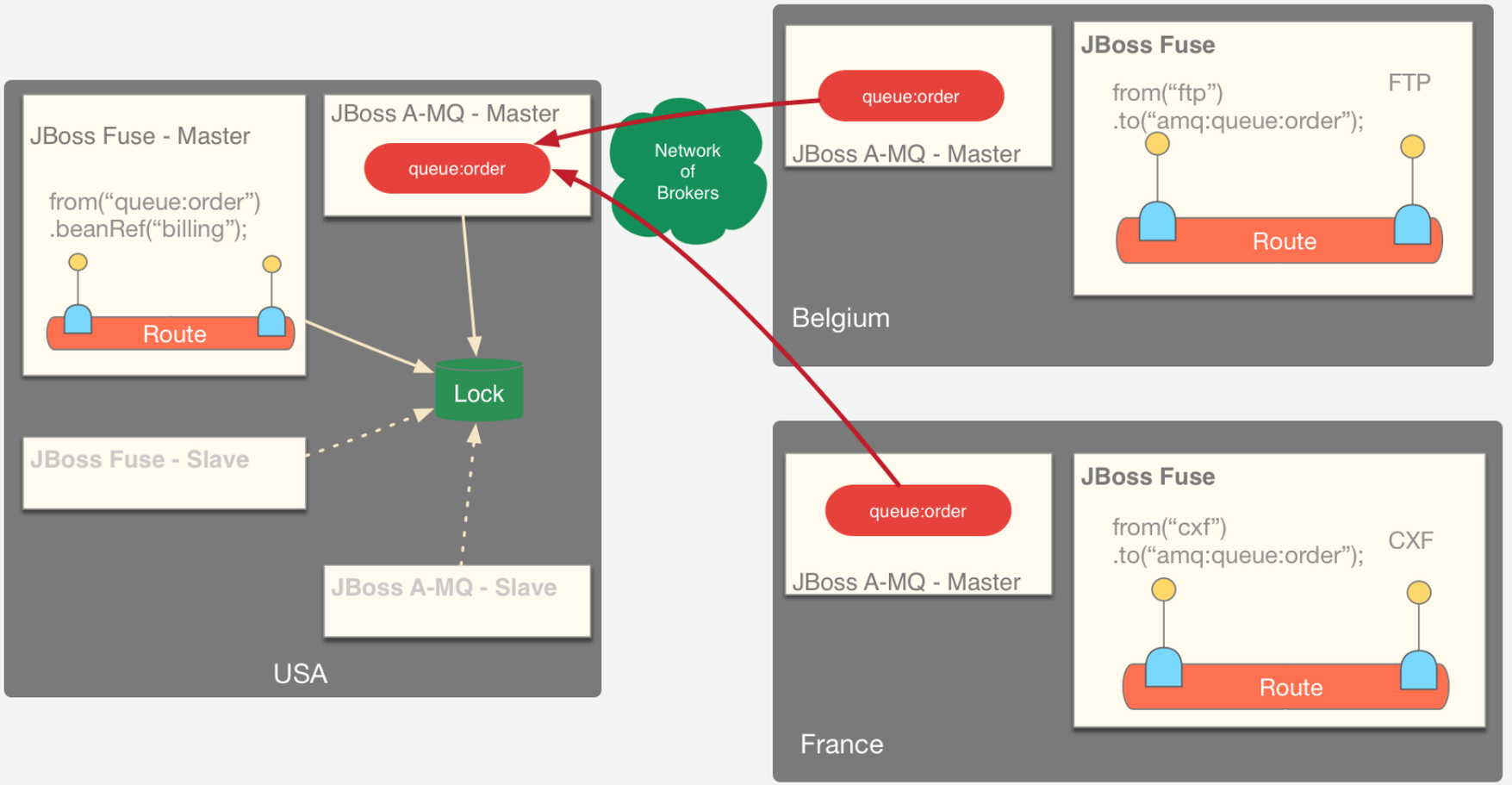
- Created when one broker establishes network connection to another broker:



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



Big picture


JBoss Fuse - Master
from("queue:order")
.beanRef("billing");
Route
JBoss A-MQ - Master
queue:order
Lock
JBoss Fuse - Slave
JBoss A-MQ - Slave
USA
Network
of
Brokers
queue:order
JBoss A-MQ - Master
JBoss Fuse
from("ftp")
.to("amq:queue:order");
FTP
Route
Belgium
queue:order
JBoss A-MQ - Master
JBoss Fuse
from("cxf")
.to("amq:queue:order");
CXF
Route
France

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

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