Integration in the Cloud

iPaaS with Apache Camel

Stan Lewis - Principal Software Engineer, Red Hat

About me...

A lead developer of the hawtio project
Primary developer of the Fuse Management Console

(infrequent) committer on Apache Camel

really just mostly do javascript/css hackery these days

Topics

- What is an iPaas?
- Fabric
  - mq-fabric
  - fabric-camel
  - and other stuff...
- hawtio

What is an iPaaS?

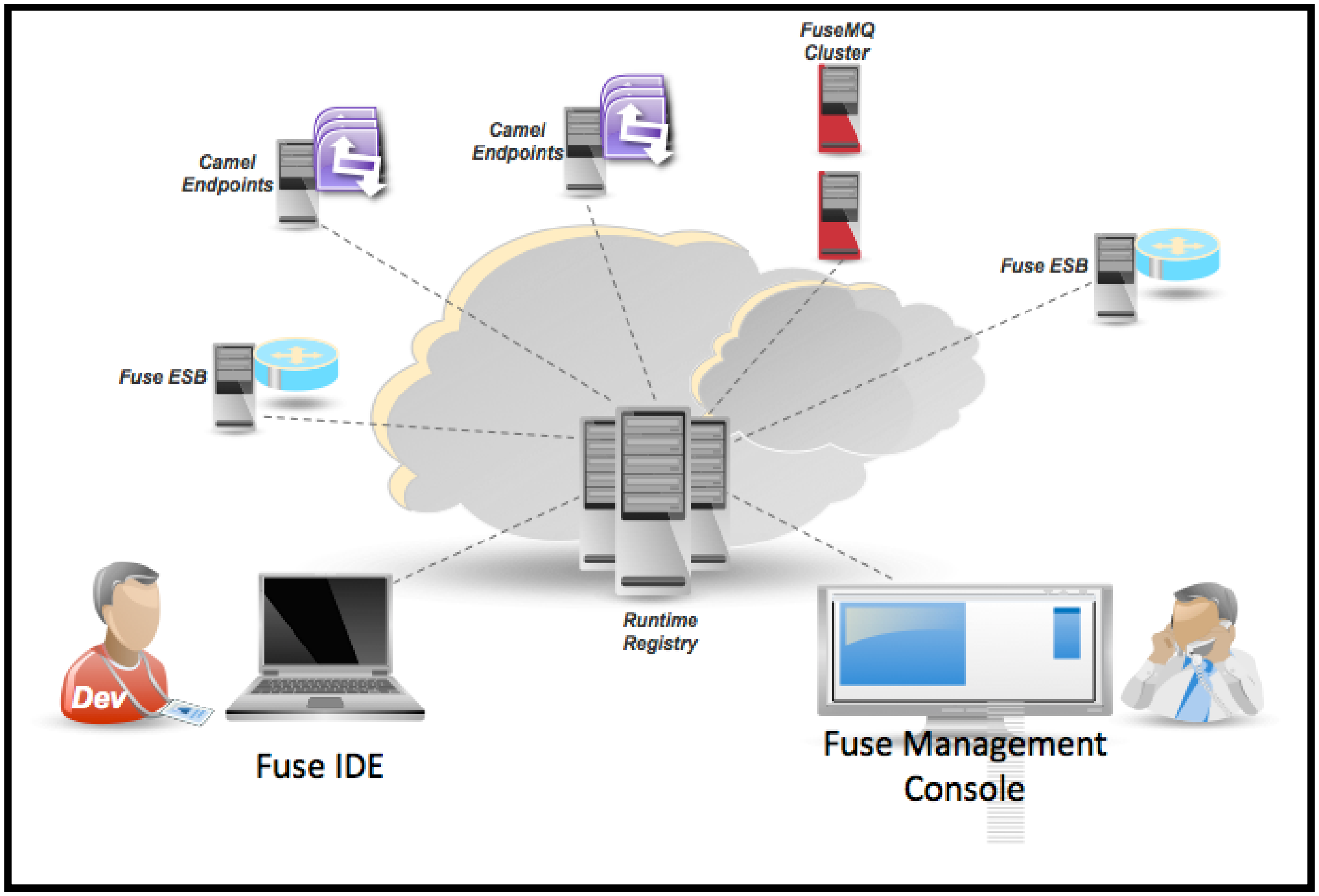
Cloudy-ish Integration

You kind of want:

- On-premises and Cloud-based hosting of integration services
- A set of connectors to integrate existing systems
  - Little or no custom development
- No need to deploy application nodes manually
- Some method of managing and monitoring integration services

Fabric + Camel + ActiveMQ

are an open source integration platform



though we'll mostly focus on Fabric and Camel

Fabric

- Is a central registry that stores configuration and runtime information used by all containers
- Configuration
  - Versions
  - Profiles
- Runtime
  - Clusters
  - Provisioning Status
- Can push Apache Karaf-based runtimes to hosts
- Most importantly, Fabric is a great environment to host Camel routes

Fabric Integration Points - ActiveMQ

Brokers can register themselves into the fabric registry as a cluster

requires the mq-fabric feature and some configuration:

```
connectors=openwire
config=zk\:/fabric/configs/versions/1.0/profiles/mq-base/broker.xml
group=default
standby.pool=default
```

You can use the "mq-create" command to generate initial configuration:

```
# Create a 3 broker master/slave cluster
mq-create --create-container broker1,broker2,broker3 hq-broker
```

or:

```
# Create a network of two broker clusters in two groups "group1" and "gr
mq-create --group us-east --networks us-west --create-container us-east1
mq-create --group us-west --networks us-east --create-container us-west1
```

Fabric Integration Points - ActiveMQ (2)

Using Fabric discovery clients can easily connect to brokers running in a fabric

Just make sure the mq-client jar is in your classpath

Or ensure the mq-fabric feature is installed in the container

```
ActiveMQConnectionFactory connectionFactory =
  new ActiveMQConnectionFactory("discovery:(fabric:<group name>)");
```

Clients outside of fabric will need to know the Zookeeper URL:

```
java -Dzookeeper.url=host1:2181,host2:2181,host3:2181 org.foo.MyAwesomeJ
```

Fabric Integration Points - Camel

Already Fabric ready when using ActiveMQ or JMS endpoints in Camel

Use the fabric-camel component to advertise socket or HTTP endpoints in the Fabric registry

```
from("fabric:MyGroup:jetty:http://0.0.0.0:8282").to("bean:myBean");
```

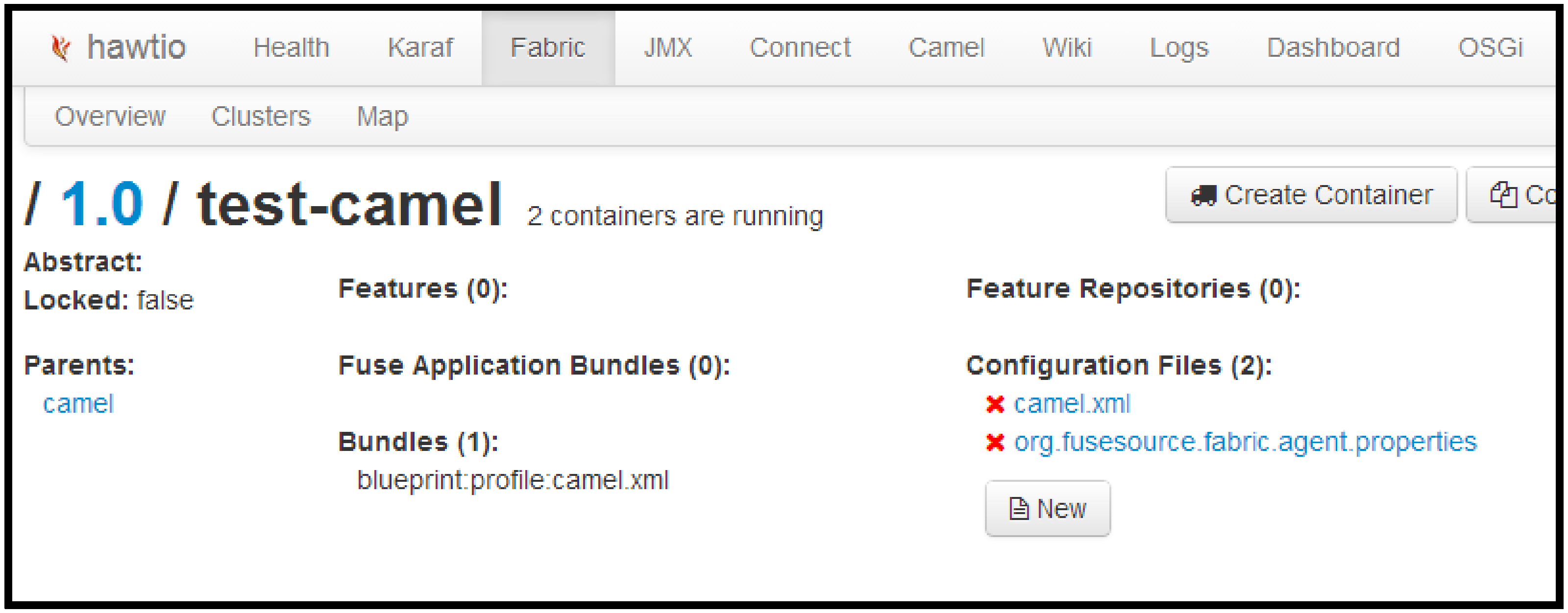
Or use fabric-camel to discover an endpoint in the Fabric registry

```
from("seda:MyQueue").to("fabric:MyGroup");
```

Fabric Integration Points
Camel, Blueprint, & Spring

Previously blueprint & spring XML files had to be put into jars

Now Fabric supports deploying a blueprint or file directly to a profile



/ 1.0 / test-camel 2 containers are running

Abstract:
Locked: false

Parents:
camel

Features (0):

Fuse Application Bundles (0):

Bundles (1):
blueprint:profile:camel.xml

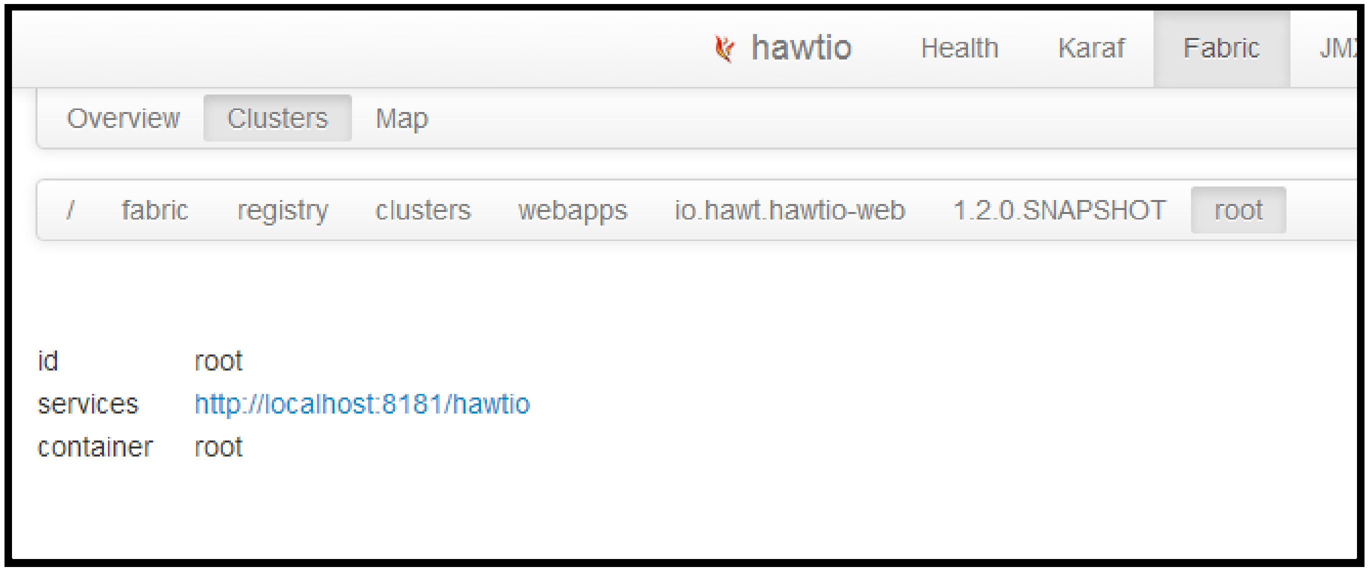
Feature Repositories (0):

Configuration Files (2):
✗ camel.xml
✗ org.fusesource.fabric.agent.properties

Fabric Integration Points - Pax Web

WAR files deployed into fabric are now automatically registered in Zookeeper

Allows easy discovery of web service endpoints



id	root
services	http://localhost:8181/hawtio
container	root

We use this in hawtio for example to connect to remote containers via Jolokia

Fabric Partition

- Provides the ability to define a task
- A task is partitioned and distributed to Fabric containers
- Each task can be associated with multiple partitions

The main use case of this feature is to generate profiles dynamically based on a template

Fabric Partition and Camel

example-camel-template (should be abstract!)

```
# org.fusesource.fabric.agent.properties.mvel
parents=camel
bundle.profile.camel-@{id}=spring:profile:camel-@{id}.xml
```

```
<!-- camel-__id__.xml.mvel -->
<beans>
  <camelcontext xmlns="http://camel.apache.org/schema/spring">
    <route>
      <from uri="@{inUri}"></from>
      <to uri="log:requests"></to>
    </route>
  </camelcontext>
</beans>
```

Fabric Partition and Camel

example-camel-partition

```
# org.fusesource.fabric.partition-example.properties
id=example
task.definition=example-camel-template
partitions.path=/fabric/partition/example
balancing.policy=even
worker.type=profile-template
```

Fabric Partition and Camel

So to recap

1. Define a template profile
2. Define a partition profile
3. Assign the partition profile to containers

    Now what?

Fabric Partition and Camel

```
# create partition 1
$ zk:create /fabric/partition/example/1 \
    "{ \\"inUri\\" : \\"direct:in1\\" }"

# create partition 2
$ zk:create /fabric/partition/example/2 \
    "{ \\"inUri\\" : \\"direct:in2\\" }"
```

hawtio

What is it?

hawtio is the evolution of the Fuse Management Console

plugins for:

- jmx
- activemq
- camel
- fabric
- osgi
- dashboard
- jboss
- jetty
- jclouds
- log
- maven>
- wiki
- ...

and at this point I'm tired of slides, so let's dive in

Questions?

http://fuse.fusesource.org/fabric

http://hawt.io