redhat.

Partners & Speakers

skills matter

Venue

c2b2
middleware experts

Organiser and Sponsor

Java Microservice Development With Apache Camel

How to develop, build and deploy microservices using Apache Camel, Spring Boot and WildFly Swarm for OpenShift / Kubernetes.

James Netherton
Red Hat Senior Software Engineer
Fuse Engineering Team

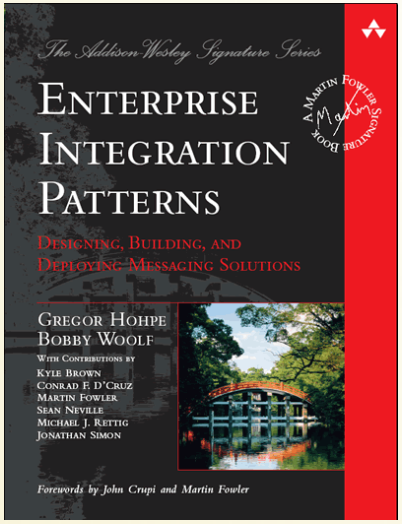
@nethertron2000 jamesnetherton

About me

- Senior Engineer at Red Hat
- Work in the JBoss Fuse engineering team
- Contribute to WildFly-Camel & Fuse Integration Services
- I like to hack on other Open Source projects
  - Camel, Fabric8, WildFly-Swarm & others

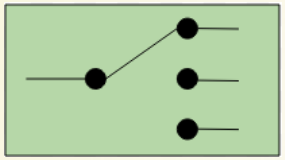
What is Apache Camel?

- Open source integration framework
- Integrate disparate systems which speak in different protocols & data formats
- Runs standalone, CDI, OSGI, Web or JavaEE containers
- Implements common Enterprise Integration Patterns

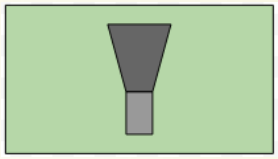


Camel EIPs

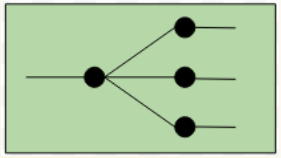
Message Router



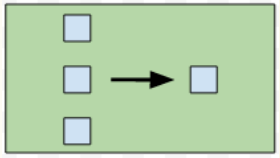
Message Filter



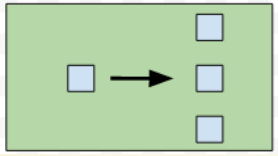
Recipient List



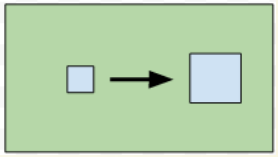
Aggregator



Splitter



Content Enricher



Camel components

Camel DSL

Java DSL

```
CamelContext camelContext = new DefaultCamelContext();
camelContext.addRoutes(new RouteBuilder() {
    @Override
    public void configure() throws Exception {
        JaxbDataFormat jaxbDataFormat = new JaxbDataFormat();
        jaxbDataFormat.setContextPath(Order.class.getPackage().getName());

        from("jms:queue:ordersIn")
        .unmarshal(jaxbDataFormat)
        .to("jpa:orders");
    }
});
```

XML

```
<camelContext>
    <route>
        <from uri="jms:queue:ordersIn"/>
        <unmarshal>
            <jaxb contextPath="org.example.com.Order"/>
        </unmarshal>
        <to uri="jpa:orders"/>
    </route>
</camelContext>
```

Microservices

"An approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms"

http://www.martinfowler.com/articles/microservices.html

Microservices

- Organised around business capabilities
- Independently replacable, upgradable, scalable
- Communicate with other services via the APIs they expose
- Resilient & designed for failure

Java Microservice frameworks

- Many Java Microservice frameworks have converged on producing executable JARs

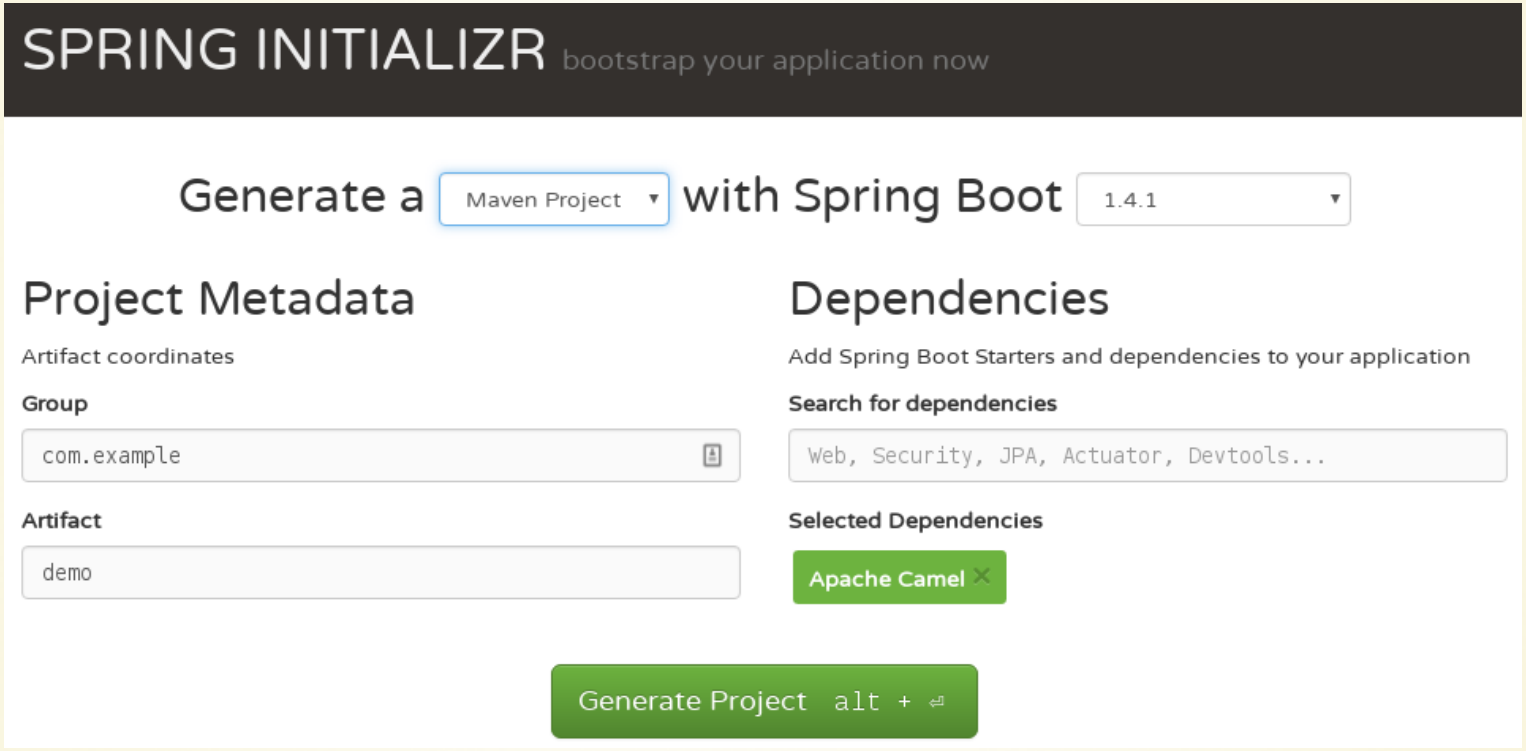
Spring Boot Camel applications

- Integration via the Spring Boot component

  http://camel.apache.org/spring-boot.html
- Configure Camel via application.properties (or YAML)
- Generate Camel apps with the Spring initializr

  http://start.spring.io/

Spring Boot Camel applications



Spring Boot Camel applications

- New in Camel 2.18!
  - Camel Spring Boot BOM
  - Camel component Spring Boot starter dependencies
  - Camel health checks via Spring Boot Actuator
  - Component properties support in application.yml with IDE auto-complete support

Spring Boot Camel applications

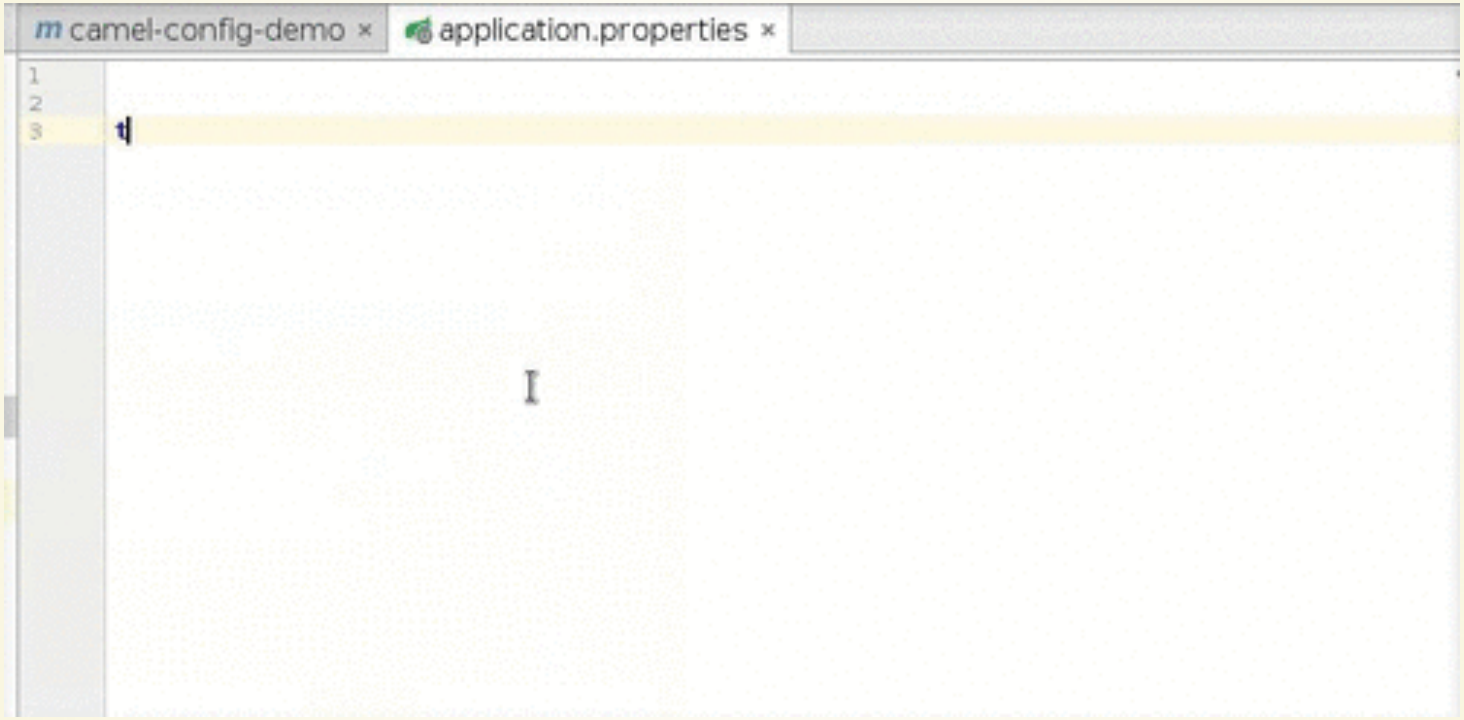
```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.apache.camel</groupId>
      <artifactId>camel-spring-boot-bom</artifactId>
      <version>2.18.0</version>
      <scope>import</scope>
      <type>pom</type>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>spring-boot-starter-undertow</artifactId>
  </dependency>
</dependencies>

<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <executions>
    <execution>
      <goals>
        <goal>repackage</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Spring Boot Camel applications

```
@SpringBootApplication
public class MyCamelSpringBootApplication {

    public static void main(String[] args) {
        SpringApplication.run(MyCamelSpringBootApplication.class, args);
    }
}
```

```
@Component
public class HelloWorldRouteBuilder extends RouteBuilder {

    public void configure() throws Exception {
        from("undertow:http://0.0.0.0:8080/hello")
        .setBody(constant("Hello World!"))
    }
}
```

Spring Boot Camel applications

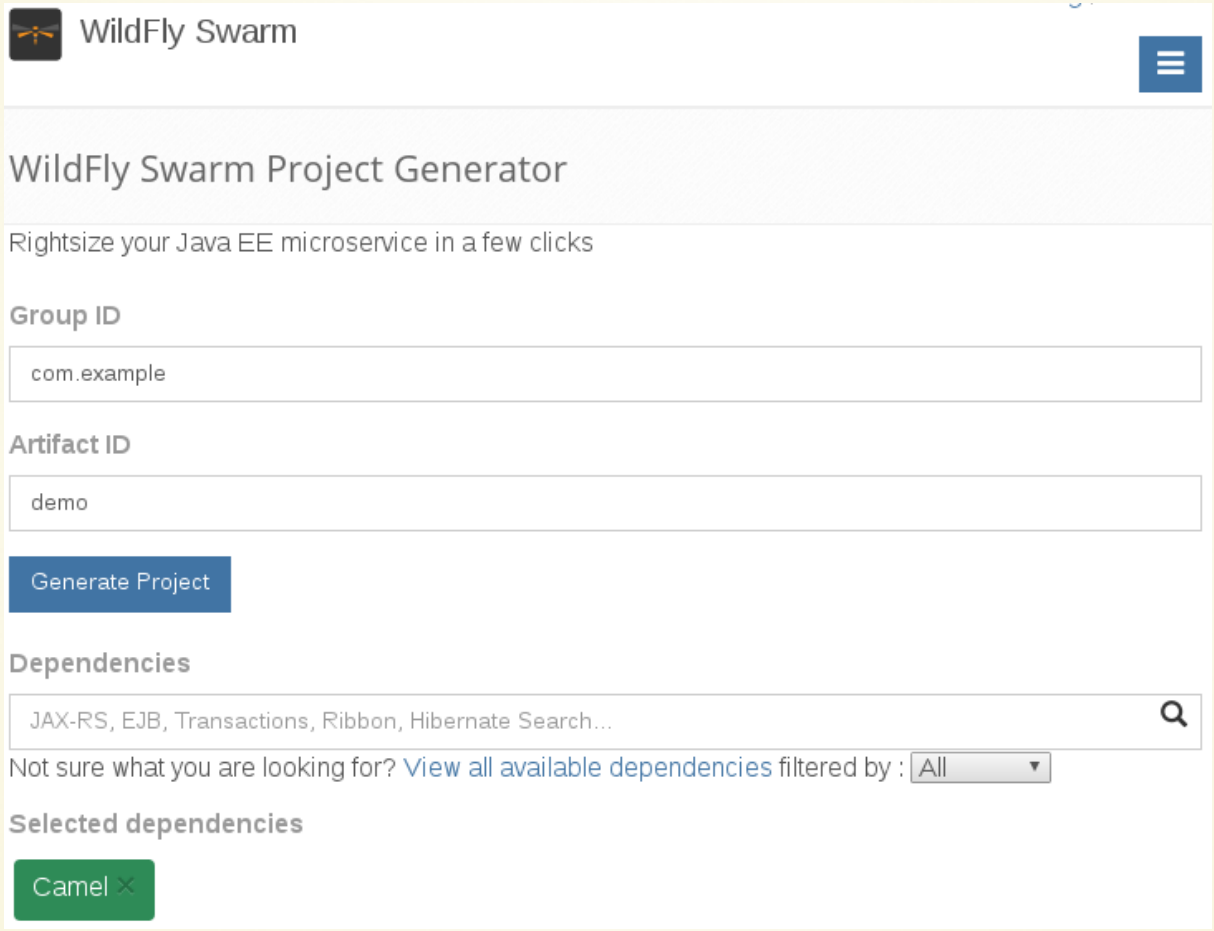


https://www.nicolaferraro.me/2016/09/25/apache-camel-meets-spring-boot/

WildFly Swarm Camel applications

- Utilises the WildFly-Camel subsystem
  https://github.com/wildfly-extras/wildfly-camel
- Use Camel APIs in your JavaEE Microservices
- Generate Camel apps with the WildFly Swarm generator
  http://wildfly-swarm.io/generator/

WildFly Swarm Camel applications



WildFly Swarm Camel applications

```
<dependencyManagement>
  <dependencies>
      <dependency>
        <groupId>org.wildfly.swarm</groupId>
        <artifactId>bom-all</artifactId>
        <version>2016.10.0</version>
        <scope>import</scope>
        <type>pom</type>
      </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>org.wildfly.swarm</groupId>
    <artifactId>camel-undertow</artifactId>
  </dependency>
</dependencies>

<plugin>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>wildfly-swarm-plugin</artifactId>
  <version>2016.10.0</version>
  <executions>
    <execution>
      <goals>
        <goal>package</goal>
      </goals>
    </execution>
  </executions>
```

WildFly Swarm Camel applications

```
public class Main {

    public static void main(Stirng args[]) {
        Swarm swarm = new Swarm()
        swarm.fraction(new CamelCoreFraction().addRouteBuilder("rest-context",
            new RouteBuilder() {
                @Override
                public void configure() throws Exception {
                    from("undertow:http://localhost/hello")
                    .log("Hello World!");
                }
        });

        swarm.start();
    }
}
```

Camel Microservice support

- Components
  - JDBC, MongoDB, JMS, Kafka, NATS, Chronicle
- Data formats
  - REST, XML, JSON, YAML, protocol buffers
- EIPs
  - Circuit breaker
  - Load balancer
- Configuration
  - Environment variables
  - Kubernetes ConfigMap

Camel Microservice configuration

- Environment variables

```
ORDERS_SERVICE_HOST=10.0.162.149
ORDERS_SERVICE_PORT=80
```

Standard environment variable property interpolation

```
public void configure() {
    from("jms:queue:orders")
    .to("http:{{ORDERS_SERVICE_HOST}}:{{ORDERS_SERVICE_PORT}}/api");
}
```

Service environment variable property interpolation

```
public void configure() {
    from("jms:queue:orders")
    .to("http:{{service:ORDERS}/api");
}
```

Camel Microservice configuration

Kubernetes ConfigMap

- Separates configuration from the Docker image
- Can be used to provide environment variables or can be read from the file system

Camel Microservice configuration

ConfigMap with Spring Cloud Kubernetes

- Specify Spring Boot application config in ConfigMaps
- Supports app reloading in response to ConfigMap updates

https://github.com/fabric8io/spring-cloud-kubernetes

Camel Microservice build & deployment

Typical use case workflow

- We want to:
  - Build the application
  - Generate a Docker image composed of our runnable JAR
  - Run the Docker image in a clustered environment like Kubernetes or OpenShift

Camel Microservice build & deployment

Which usually means:

- Configuring a Maven plugin to build Docker images
- Specifying desired Kubernetes resources in YAML / JSON
- Apply Kubernetes resources to the cluster

Camel Microservice build & deployment

Fabric8 Maven Plugin (f-m-p) to the rescue!

- Uses generators and encrichers to make opinionated decisions on how an app is built
- Generates Docker images & Kubernetes resources
- Highly extendable and configurable

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>fabric8-maven-plugin</artifactId>
  <executions>
    <execution>
      <id>fmp</id>
      <goals>
        <goal>resource</goal>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Camel Microservice build & deployment

- Basic workflow:
  - Generate resources - fabric8:resource
  - Build docker images - fabric8:build
  - Deploy changes - fabric8:deploy
- Also supports OpenShift S2I workflow

Camel Microservice build & deployment

- Generators
  - Spring Boot
  - WildFly Swarm
  - Web Application
  - Karaf
- Determines the type of application being built and configures appropriate defaults:
  - Docker base image
  - Health & readiness checks
  - Exposed ports, startup command and more

Camel Microservice build & deployment

There's more!

Tail pod logs	fabric8:log
In-container debugging	fabric8:debug
Watch for changes	fabric8:watch
Start application	fabric8:start
Stop application	fabric8:stop
Delete application	fabric8:undeploy

Camel Microservice testing

- Camel has a comprehensive test framework for various technologies:
  - Spring
  - CDI
  - OSGi

http://camel.apache.org/testing.html

Camel Microservice testing

Fabric8 Arquillian

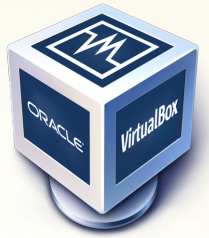
- Integration testing on OpenShift & Kubernetes
- Provision & orchestrate containers
- Assertj based DSL to verify the expected resources are running

https://github.com/fabric8io/fabric8/tree/master/components/fabric8-arquillian

Camel Microservice testing

```
@RunWith(Arquillian.class)
@RunAsClient
public class KubernetesIntegrationTest {

    @ArquillianResource
    KubernetesClient client;

    @Test
    public void testAppProvisionsRunningPods() throws Exception {
        assertThat(client).deployments().pods().isPodReadyForPeriod();
    }
}
```

Setting up local clusters for development



RED HAT®
CONTAINER
DEVELOPMENT KIT

- Or you could just run natively
- Or better still...

Setting up local clusters for development

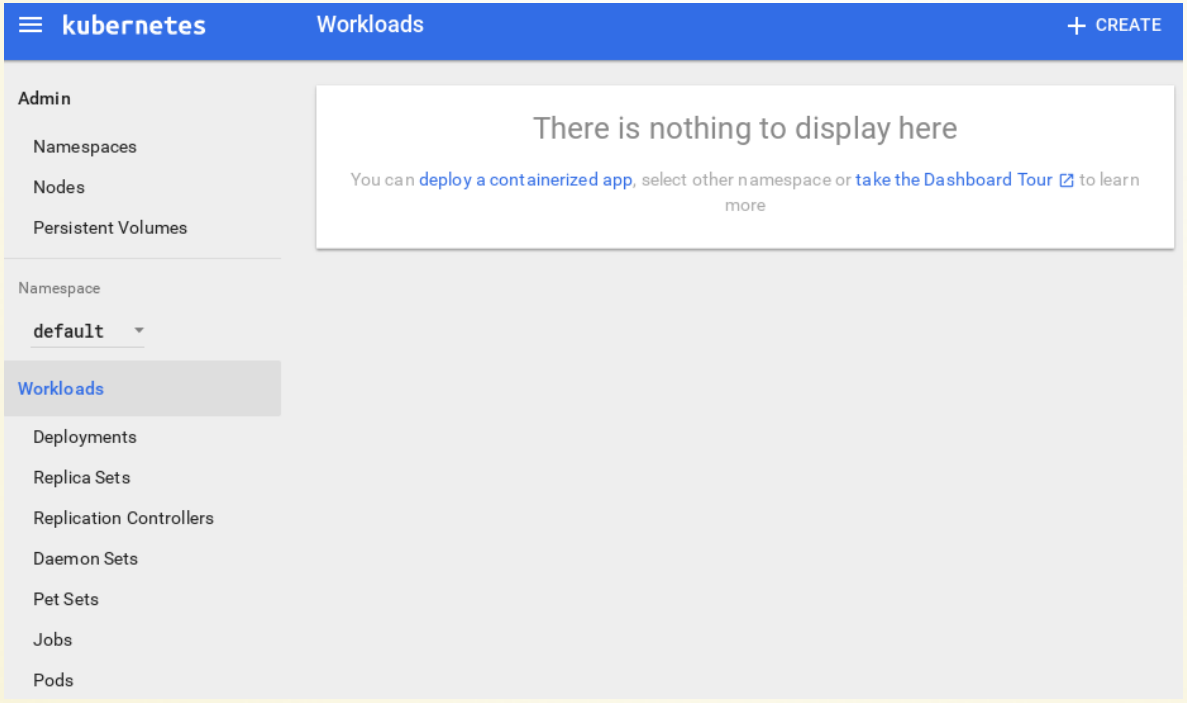
- Minikube

  https://github.com/kubernetes/minikube
- Minishift

  https://github.com/MiniShift/minishift
- Runs a single node local cluster within a lightweight VM

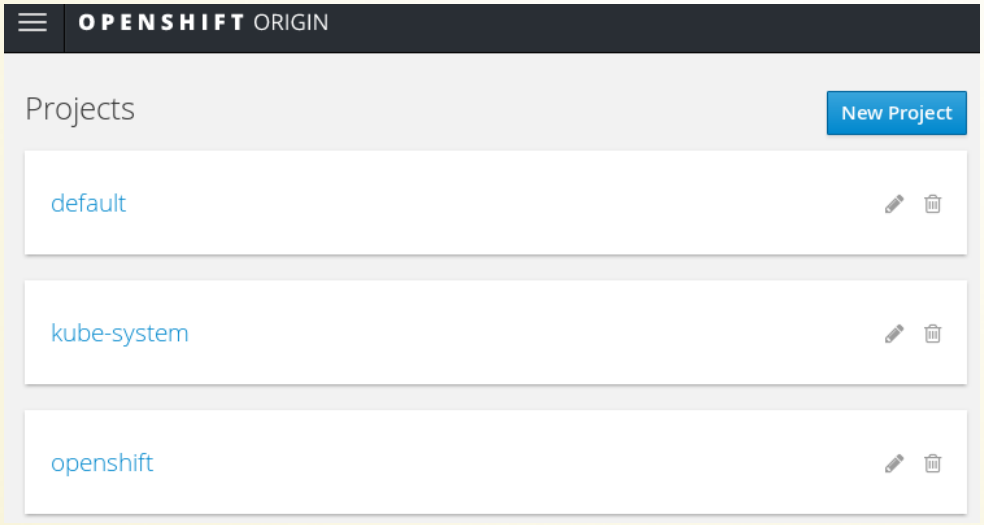
Setting up local clusters for development

```
$ minikube start

Starting local Kubernetes cluster...
Kubectl is now configured to use the cluster.


$ minikube dashboard

Opening kubernetes dashboard in default browser...
```



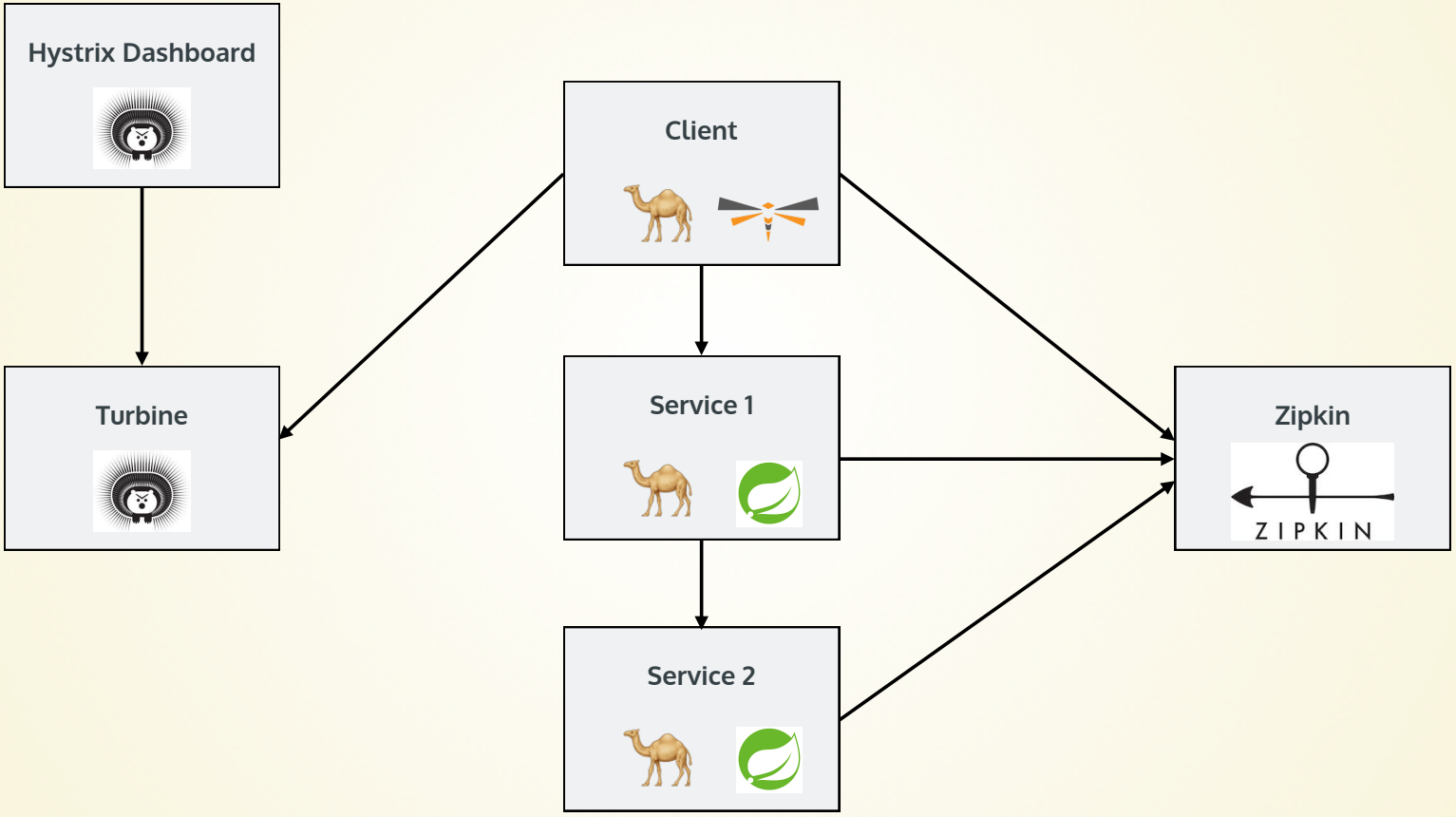
Setting up local clusters for development

```
$ minishift start --insecure-registry=my-insecure-registry.org \
                  --deploy-router=true

Starting local OpenShift cluster...
oc is now configured to use the cluster.
Run this command to use the cluster:
oc login --username=admin --password=admin

$ minishift console

Opening OpenShift console in default browser...
```



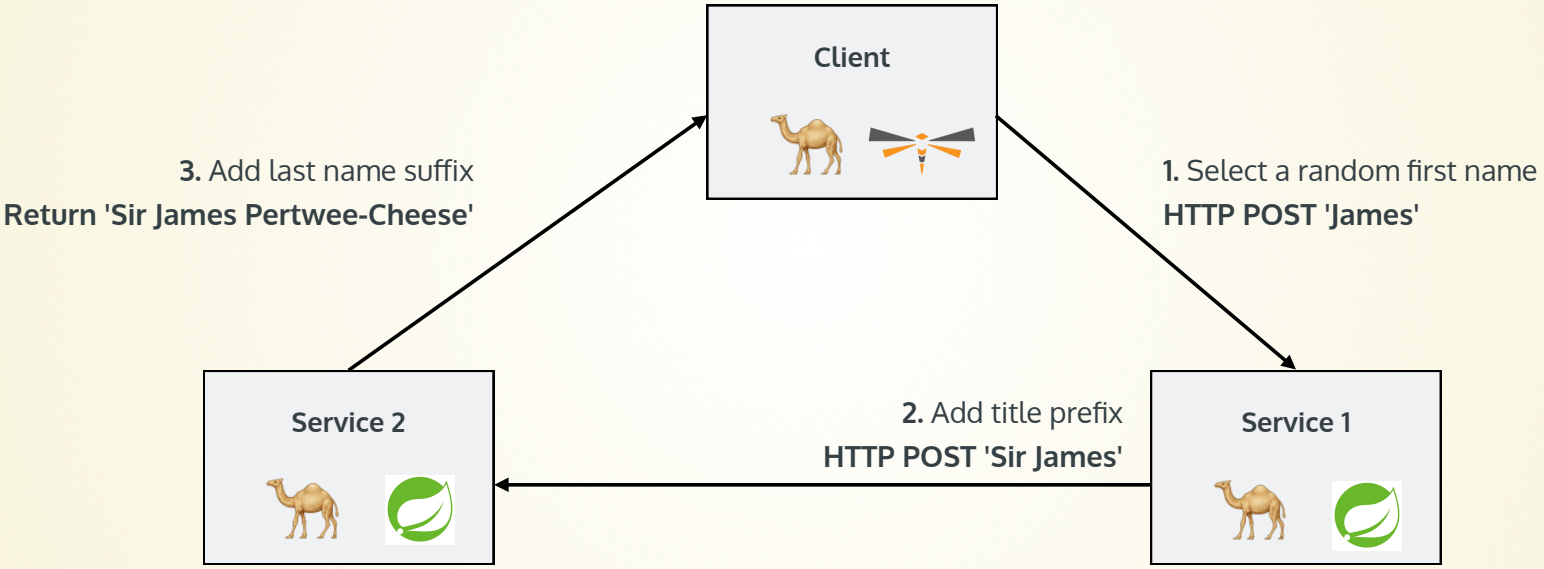
DEMO TIME

Camel Microservice demo



Camel Microservice demo

Name generator



Camel Microservice demo

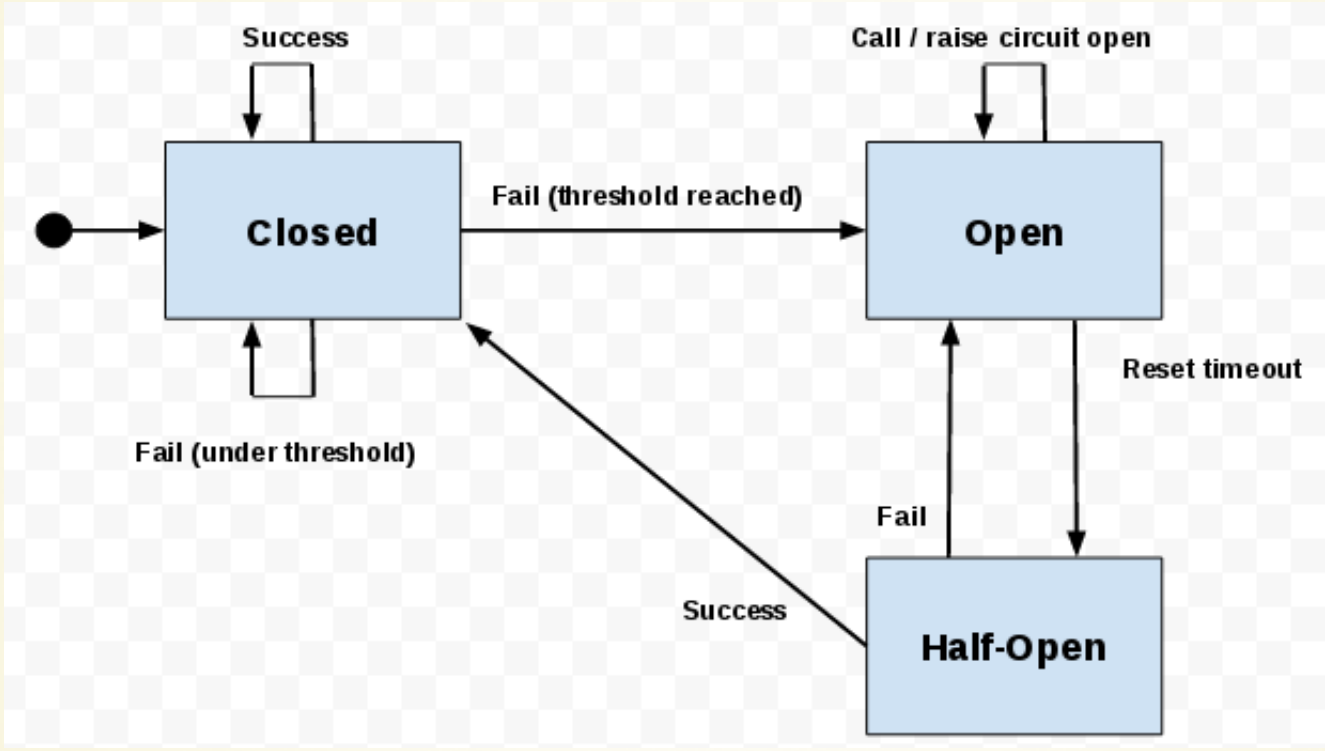
Building for resilience

- Things ***will*** fail
- Camel Hystrix component + circuit breakers can help:

"*Wrap a protected function call in a circuit breaker object, which monitors for failures. Once the failures reach a certain threshold, the circuit breaker trips, and all further calls to the circuit breaker return with an error, without the protected call being made at all*"

http://martinfowler.com/bliki/CircuitBreaker.html

Camel Microservice demo



Camel Microservice demo

```
public void configure() throws Exception {
    from("direct:start")
    .hystrix()
        .hystrixConfiguration()
            .executionTimeoutInMilliseconds(5000)
        .end()
        .to("jetty:http://mydomain.com/some-endpoint")
        .onFallback()
            .setBody(constant("I'm not available right now"))
        .end()
}
```

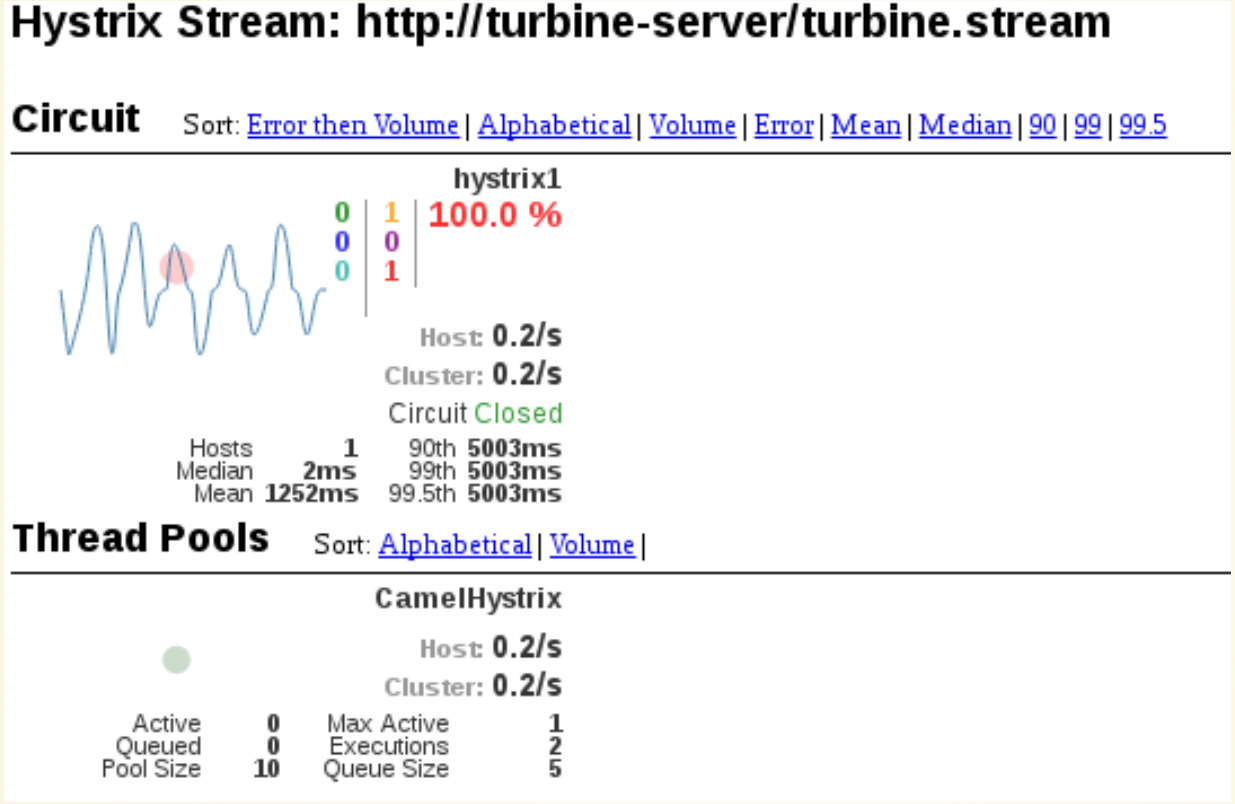
Camel Microservice demo

- Kubeflix
  - Kubernetes discovery for Hystrix & Turbine
  - Docker images for Turbine and Hystrix Dashboard
  - Discovery works via a simple label annotations

    https://github.com/fabric8io/kubeflix

```
metadata:
  labels:
    hystrix.enabled: "true"
```

Camel Microservice demo

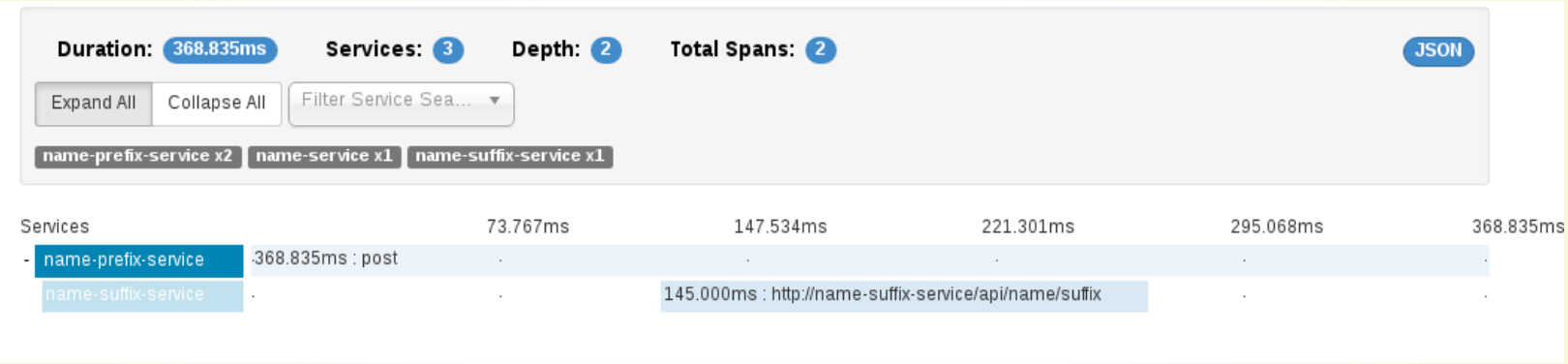


Camel Microservice demo

Distributed tracing

- Camel Zipkin component can capture timings for incoming & outgoing messages
- Publish trace metrics to Zipkin collector server
- View the results in Zipkin web UI

Camel Microservice demo



Links

Projects

http://camel.apache.org

https://projects.spring.io/spring-boot

http://wildfly-swarm.io

https://kubernetes.io

https://openshift.org

Examples

https://github.com/jamesnetherton/camel-microservice-demo

https://github.com/fabric8-quickstarts

https://github.com/wildfly-swarm/wildfly-swarm-examples

My ramblings

https://twitter.com/nethertron2000

https://jamesnetherton.github.io