

MICROSERVICES JOURNEY WITH APACHE CAMEL

OCTOBER 4, 2016 | ATLANTA, GA



Developing
microservices
with Apache Camel

Atlanta 2011



<https://devnexus.com>

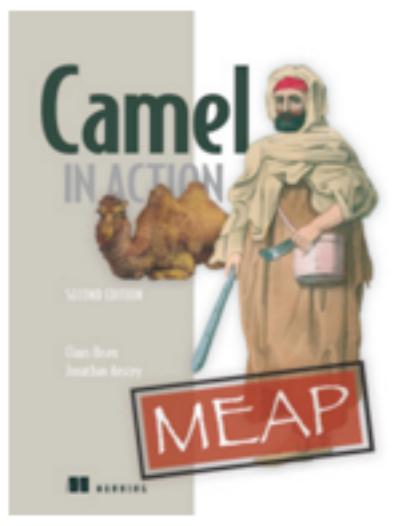
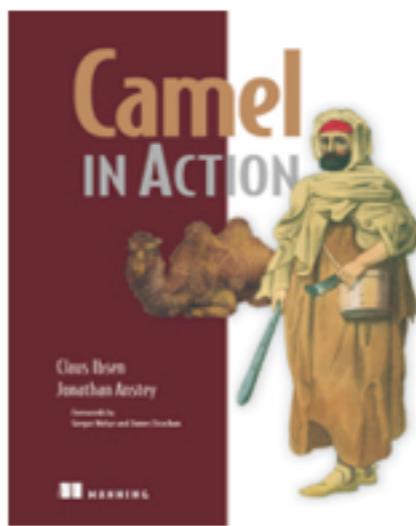
Claus Ibsen



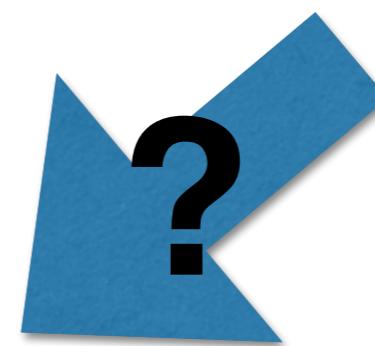
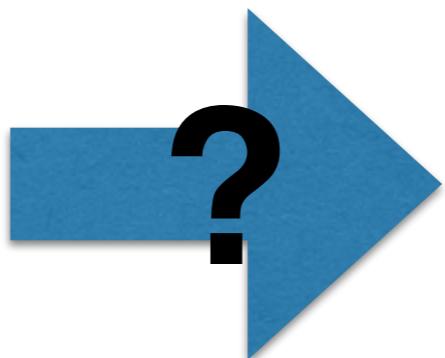
- Principal Software Engineer at Red Hat
- Apache Camel
8 years working with Camel
- Author of Camel in Action books



@davsclaus
davsclaus
davsclaus.com



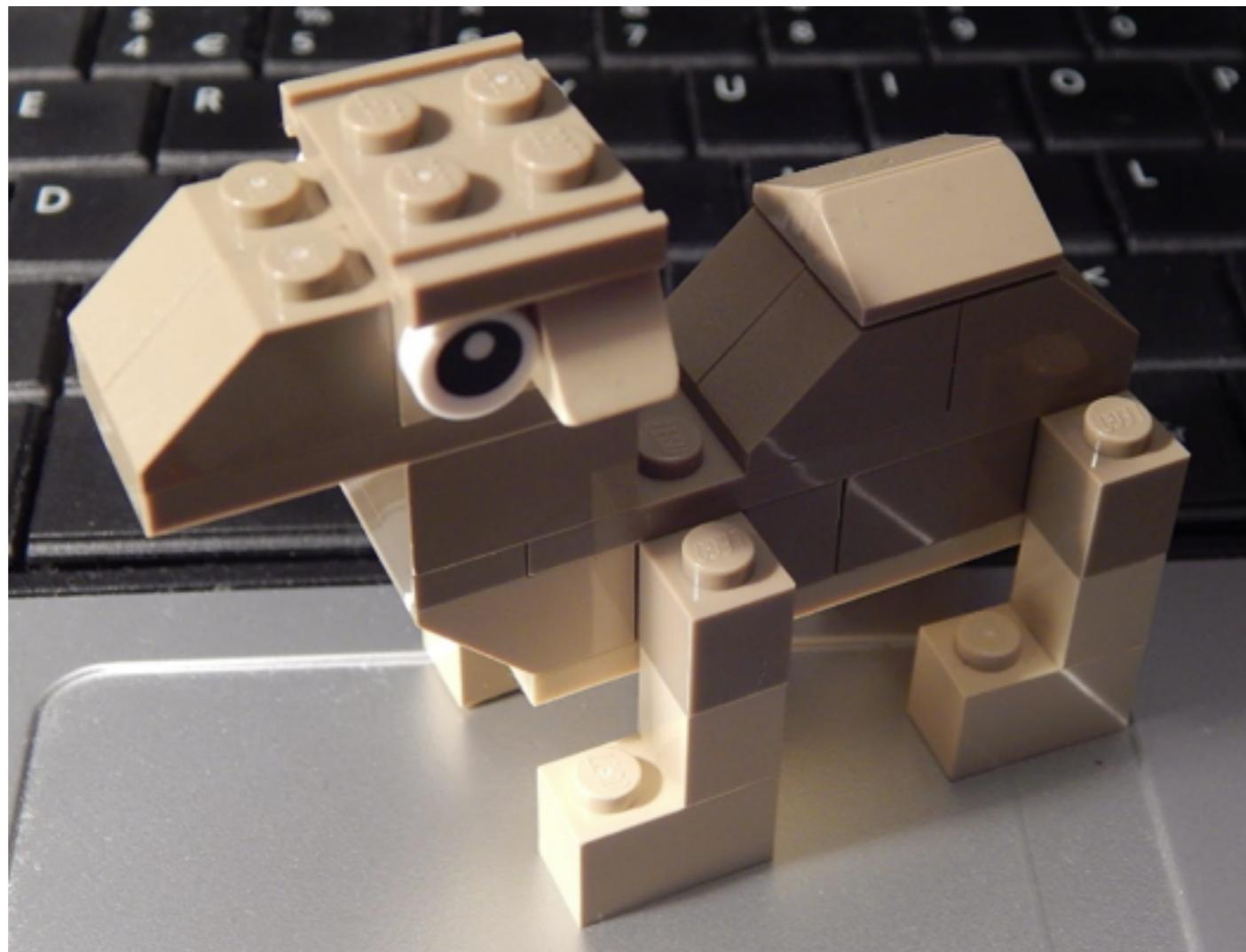
Java Developer



or



Build a Camel Demo Time



Source Code

 GitHub, Inc. [US] <https://github.com/davsclaus/fabric8-hello>

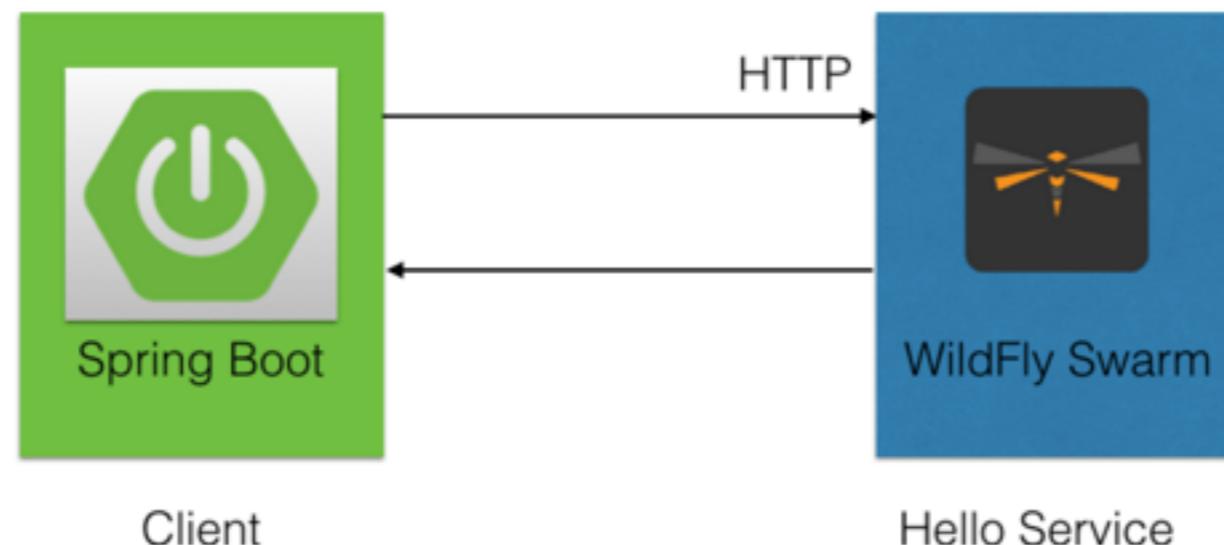
fabric8-hello

Two microservices using Spring Boot and WildFly Swarm with Camel running in kubernetes using

There are two Maven projects:

- client - Spring Boot application with Camel that triggers every 2nd second to call the hello service response.
- helloswarm - WildFly Swarm application hosting a hello service which returns a reply message

The diagram below illustrates this:



<https://github.com/davsclaus/fabric8-hello>

Hello Service

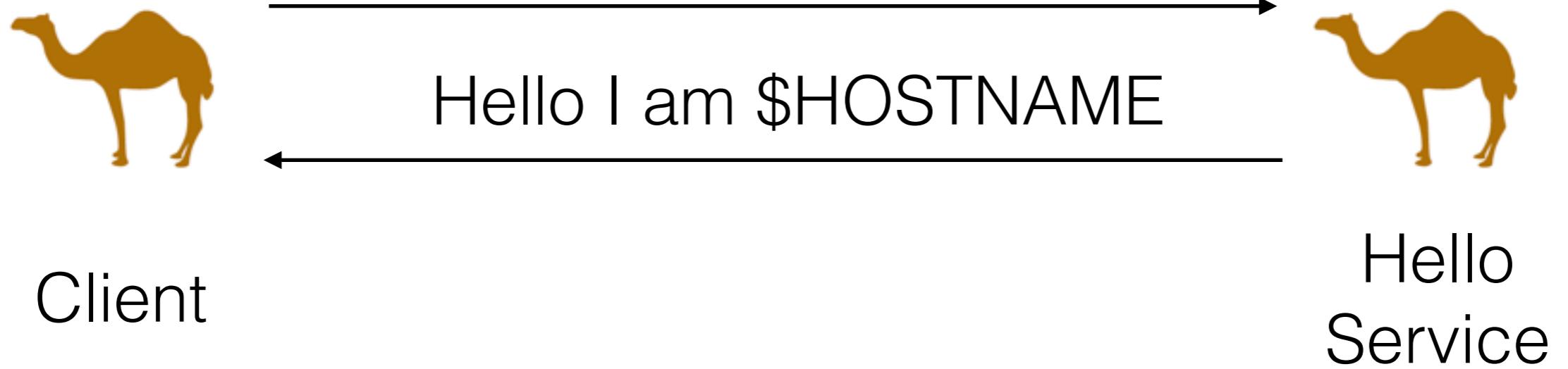


Client

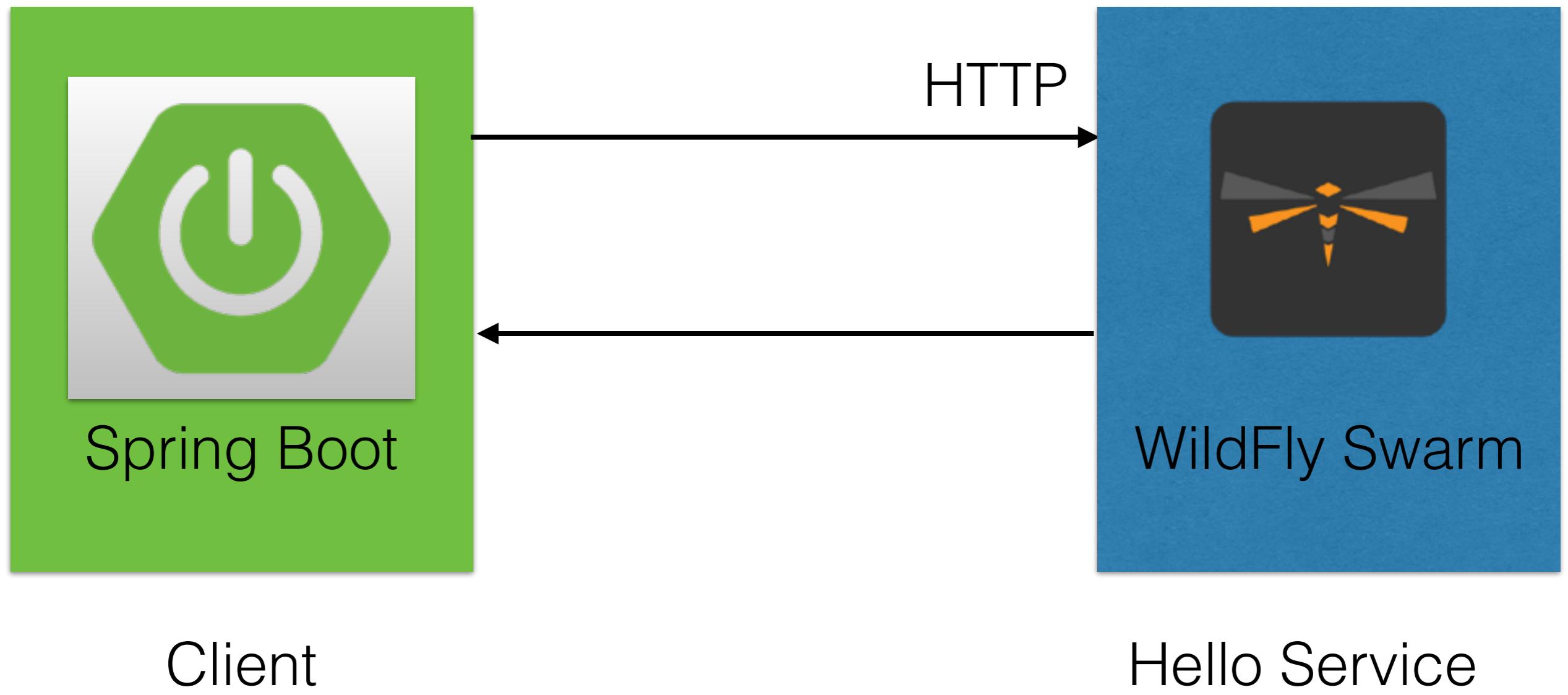


Hello
Service

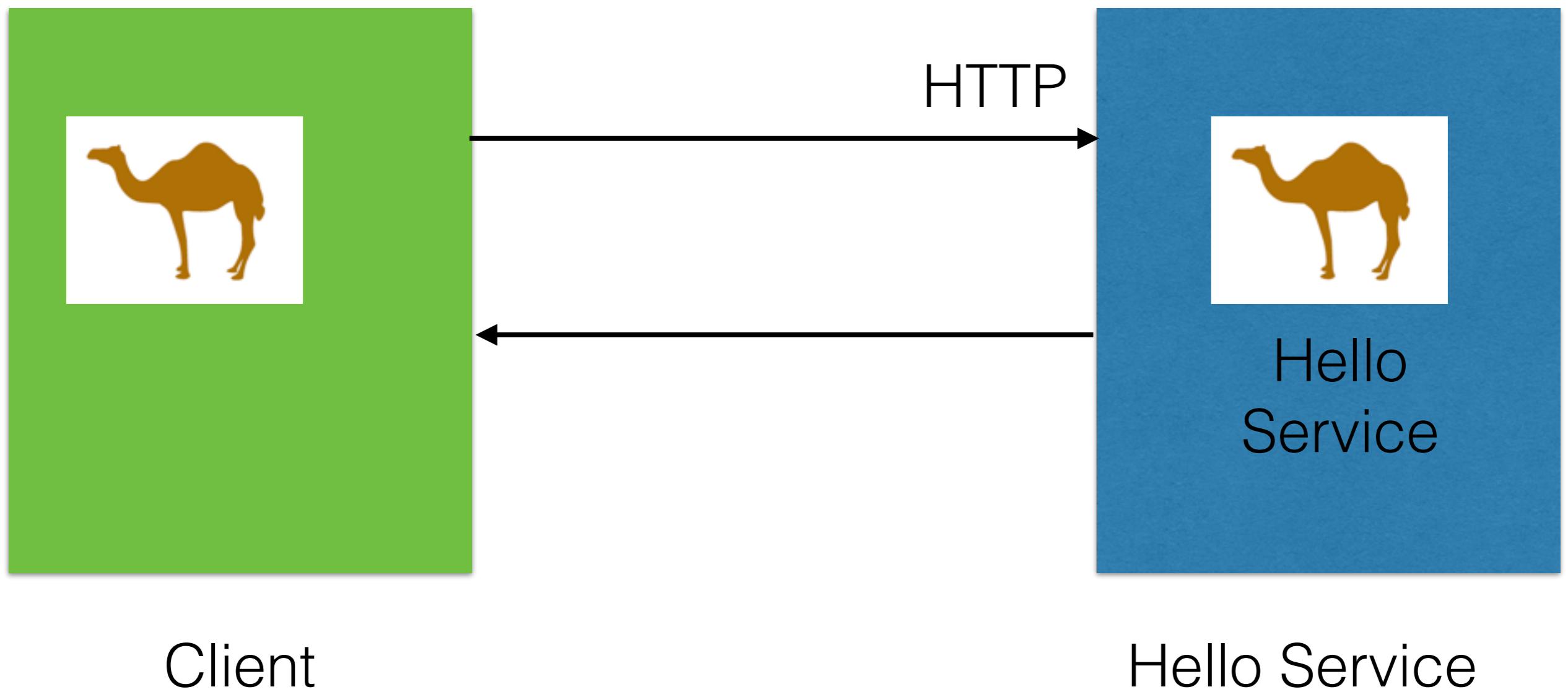
Hello Service



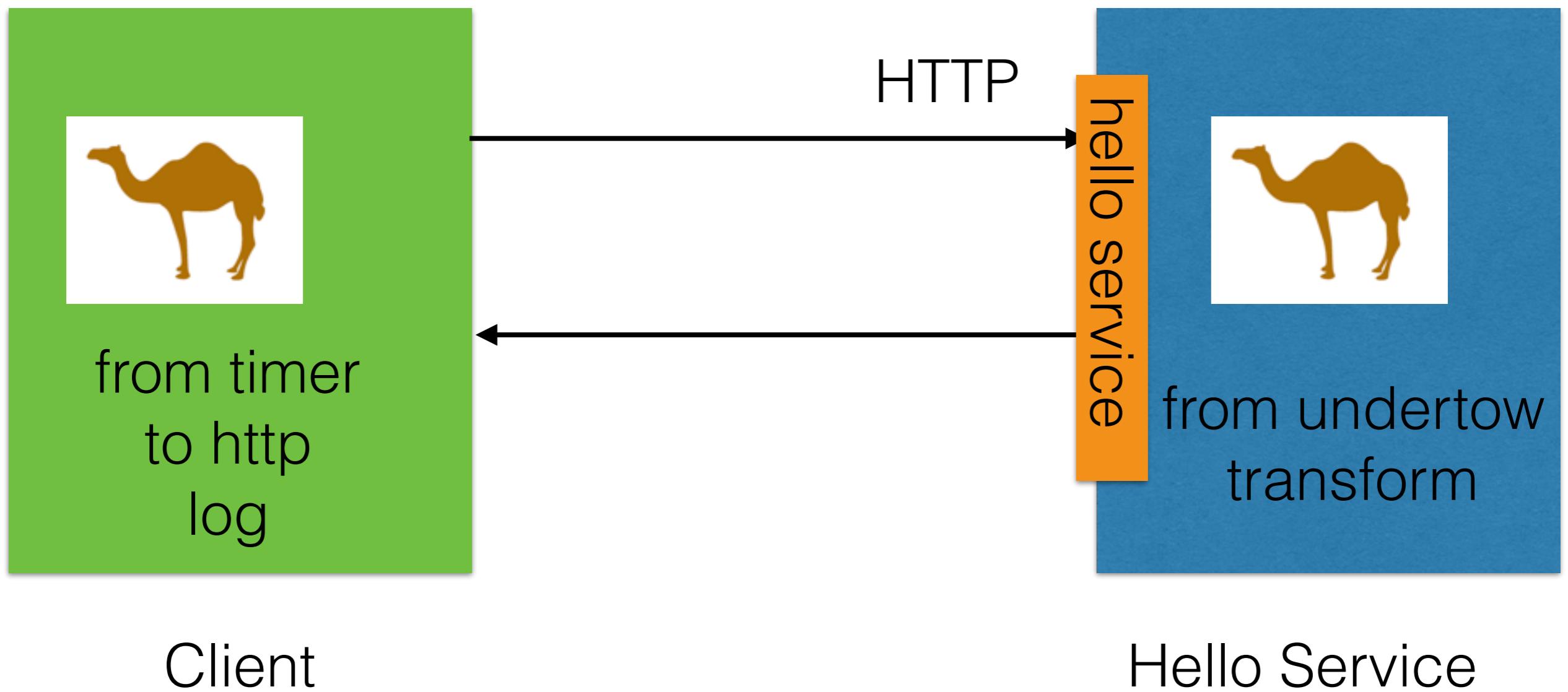
Implementation



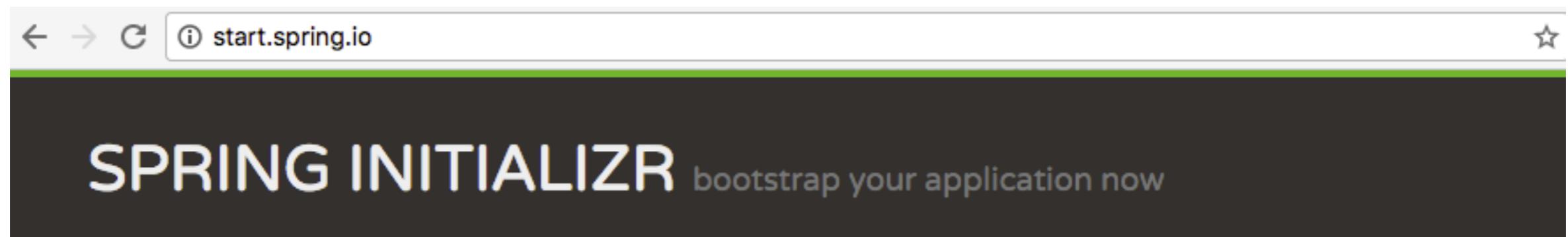
Implementation



Implementation



Spring Boot Starter



Generate a Maven Project with Spring Boot 1.4.0

Project Metadata

Artifact coordinates

Group

com.foo

Artifact

client

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Web ×

Actuator ×

Apache Camel ×

Generate Project ⌂ + ↵



Client

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer:foo?period=2000")
            .to("netty4-http:http://localhost:8080/hello")
            .log("${body}");
    }
}
```

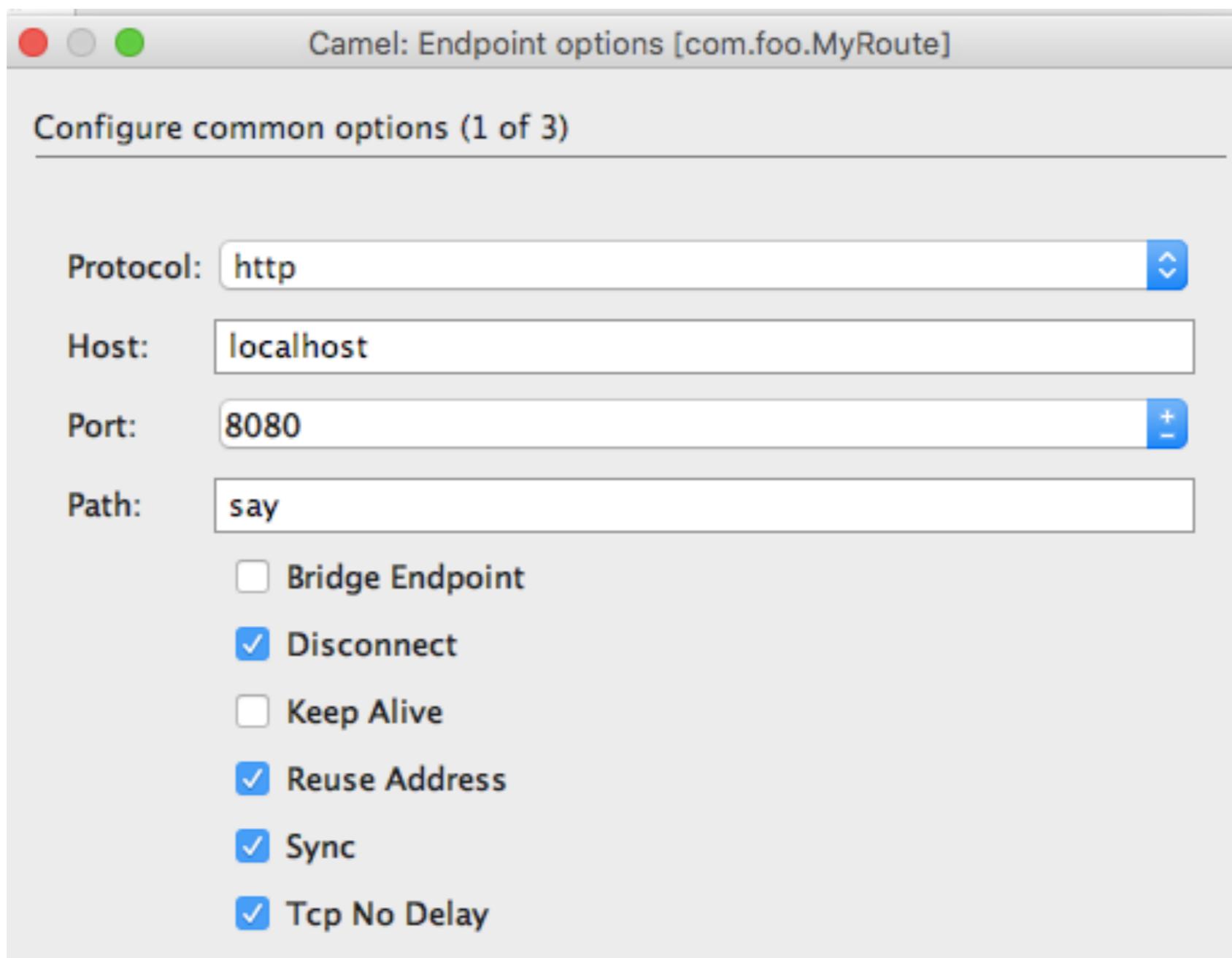
Type Safe Camel Editor

```
from("timer:foo?period=2000")
    .to("")~
```

Cursor is here

<http://fabric8.io/guide/forge.html>

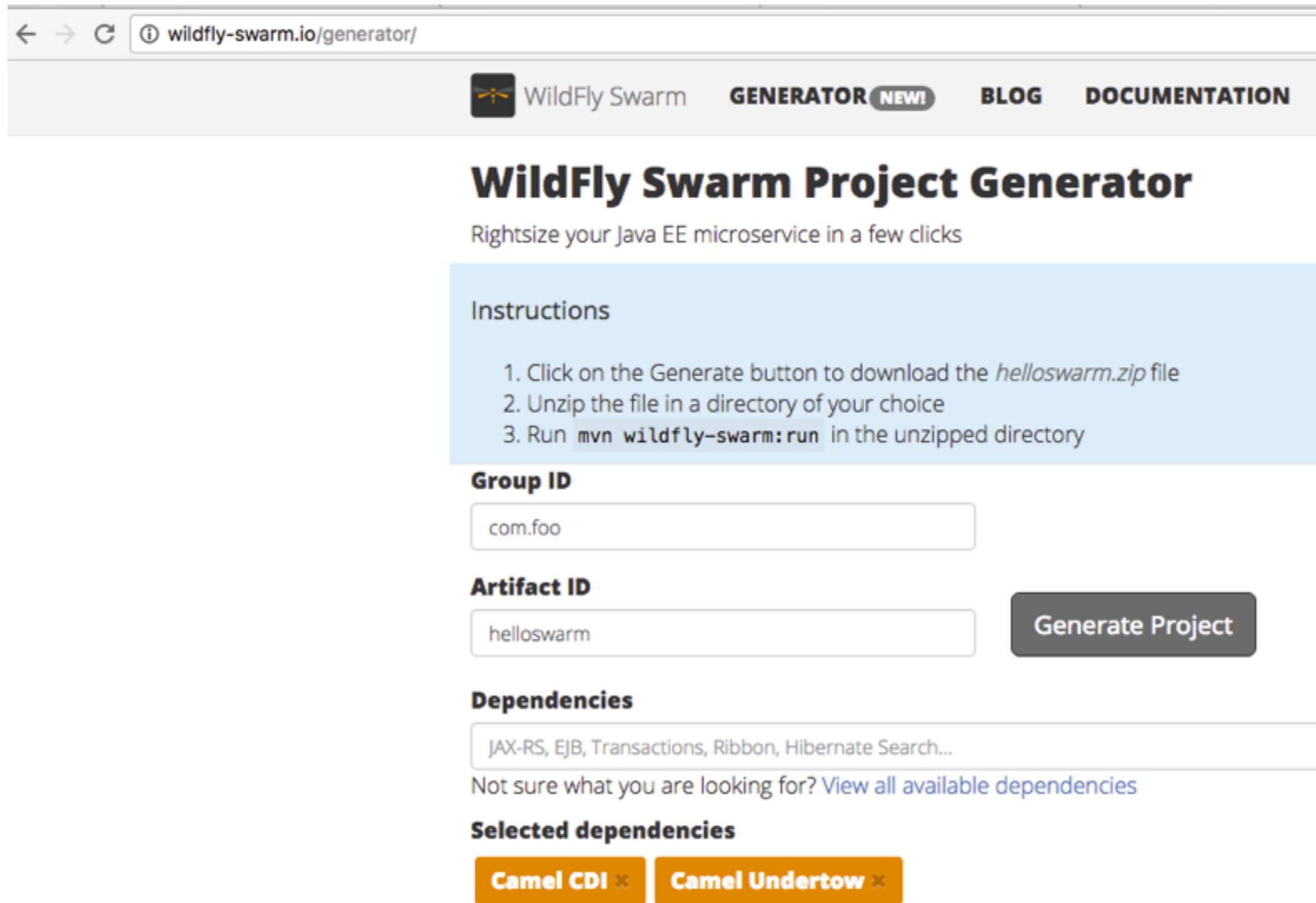
Type Safe Camel Editor



Type Safe Camel Editor

```
from("timer:foo?period=2000")
    .to("netty4-http:http:localhost:8080/say?disconnect=true&keepAlive=false")~
```

WildFly Swarm Generator



The screenshot shows the WildFly Swarm Project Generator interface. At the top, there's a header bar with navigation icons, a URL bar showing `wildfly-swarm.io/generator/`, and a logo for WildFly Swarm. Below the header, the main title is **WildFly Swarm Project Generator**. A subtitle says "Rightsize your Java EE microservice in a few clicks". A light blue box contains instructions: "1. Click on the Generate button to download the `helloswarm.zip` file
2. Unzip the file in a directory of your choice
3. Run `mvn wildfly-swarm:run` in the unzipped directory". There are input fields for "Group ID" (containing "com.foo") and "Artifact ID" (containing "helloswarm"). A large "Generate Project" button is next to the artifact ID field. Below these, a "Dependencies" section lists "JAX-RS, EJB, Transactions, Ribbon, Hibernate Search...". A link "View all available dependencies" is provided. Under "Selected dependencies", two items are listed: "Camel CDI" and "Camel Undertow".

← → ⌂ i wildfly-swarm.io/generator/

 WildFly Swarm **GENERATOR** NEW! **BLOG** **DOCUMENTATION**

WildFly Swarm Project Generator

Rightsize your Java EE microservice in a few clicks

Instructions

1. Click on the Generate button to download the `helloswarm.zip` file
2. Unzip the file in a directory of your choice
3. Run `mvn wildfly-swarm:run` in the unzipped directory

Group ID
com.foo

Artifact ID
helloswarm **Generate Project**

Dependencies

JAX-RS, EJB, Transactions, Ribbon, Hibernate Search...
Not sure what you are looking for? [View all available dependencies](#)

Selected dependencies

Camel CDI × **Camel Undertow** ×



Hello Service

```
@Singleton
public class HelloRoute extends RouteBuilder {

    @Inject
    @Uri("undertow:http://0.0.0.0:8080/say")
    private Endpoint undertow;

    @Inject
    private HelloBean hello;

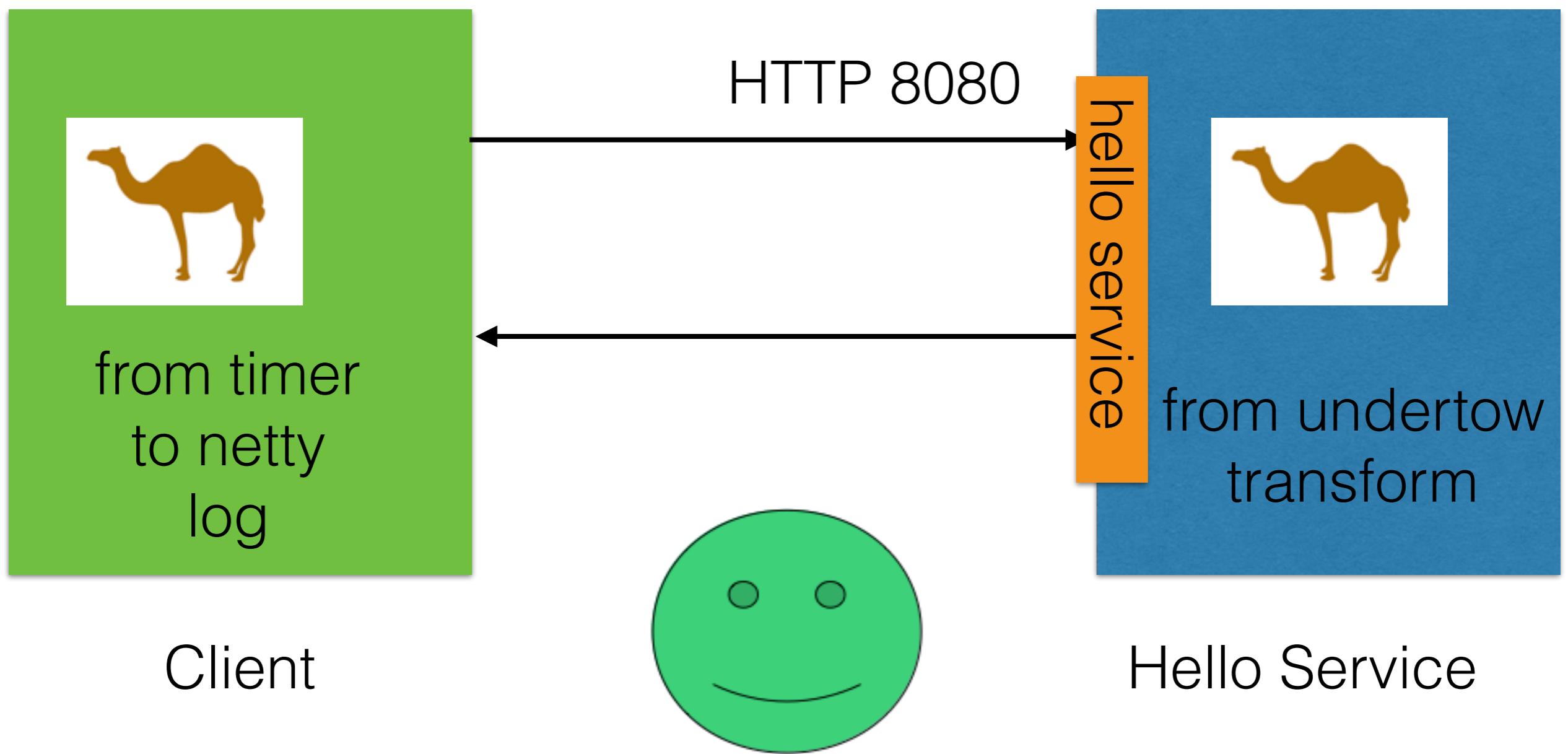
    @Override
    public void configure() throws Exception {
        from(undertow).bean(hello);
    }
}
```



Hello Service

```
@Singleton  
public class HelloBean {  
  
    public String sayHello() throws Exception {  
        return "Swarm says hello from "  
            + InetAddressUtil.getLocalHostName();  
    }  
}
```

Ready to run local



How to build Docker Image?

Maven Project

Docker Image

Docker Maven Plugin



<https://maven.fabric8.io>

Fabric8 Maven Plugin

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>fabric8-maven-plugin</artifactId>
  <version>3.1.63</version>
</plugin>
```

<https://maven.fabric8.io>

Install fabric8-maven-plugin

```
mvn io.fabric8:fabric8-maven-plugin:3.1.63:setup
```

<https://maven.fabric8.io>

Fabric8 Maven Plugin

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>fabric8-maven-plugin</artifactId>
  <version>3.1.63</version>
  <executions>
    <execution>
      <id>fmp</id>
      <goals>
        <goal>resource</goal>
        <goal>helm</goal>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Build Docker Image

```
mvn package fabric8:build
```

```
[INFO] --- fabric8-maven-plugin:3.1.32:build (fmp) @ client ---
[INFO] F8> Running in Kubernetes mode
[INFO] F8> Running generator spring-boot
[INFO] F8> Running generator java-exec
[INFO] F8> Pulling from fabric8/java-alpine-openjdk8-jdk
e110a4a17941: Pull complete
deb4805e2548: Pull complete
04712c369ba1: Pull complete
2c82593e8eb2: Downloading [=====> ] 49.54 MB/49.68 MB
58bb43a36a2e: Download complete
a2f01f9f1b00: Download complete
e6f2f9c9e249: Download complete
179ec0f1d75a: Download complete
61cbb3b63095: Download complete
5bd50883c949: Download complete
```

□

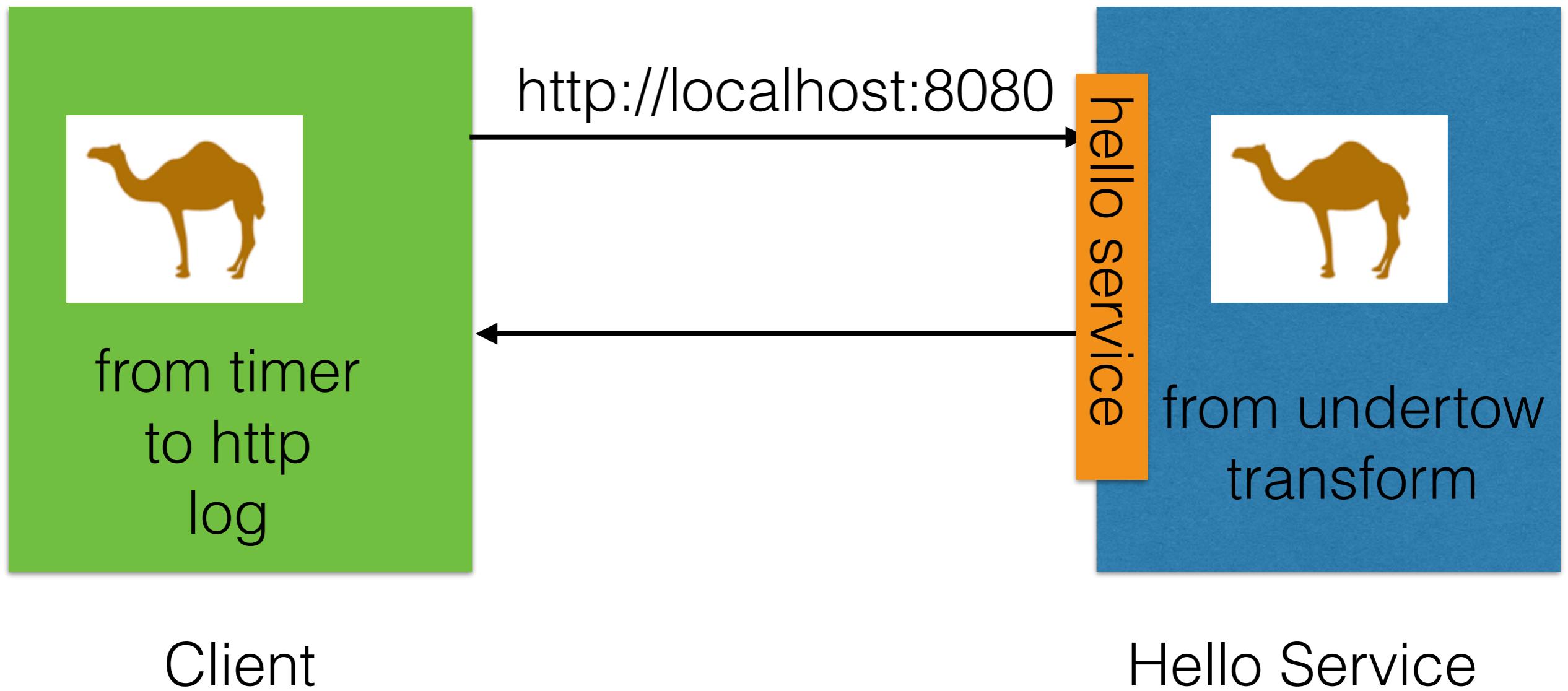
Local Docker Repository

davsclaus:/Users/	documents/workspace/client/\$ docker images	
REPOSITORY	TAG	IMAGE ID
foo/helloswarm	latest	f572db11fc2e
foo/client	latest	c16cac32bf39
openshift/origin-deployer	v1.3.0-rc1	7e0bab2a9a21
openshift/origin-node	v1.3.0-rc1	ea67ec68a53a
fabric8/exposed	latest	7e2e98c75db5
fabric8/fabric8	2.2.174	efb7bc509cb0
fabric8/java-al	latest	c9139d27b712

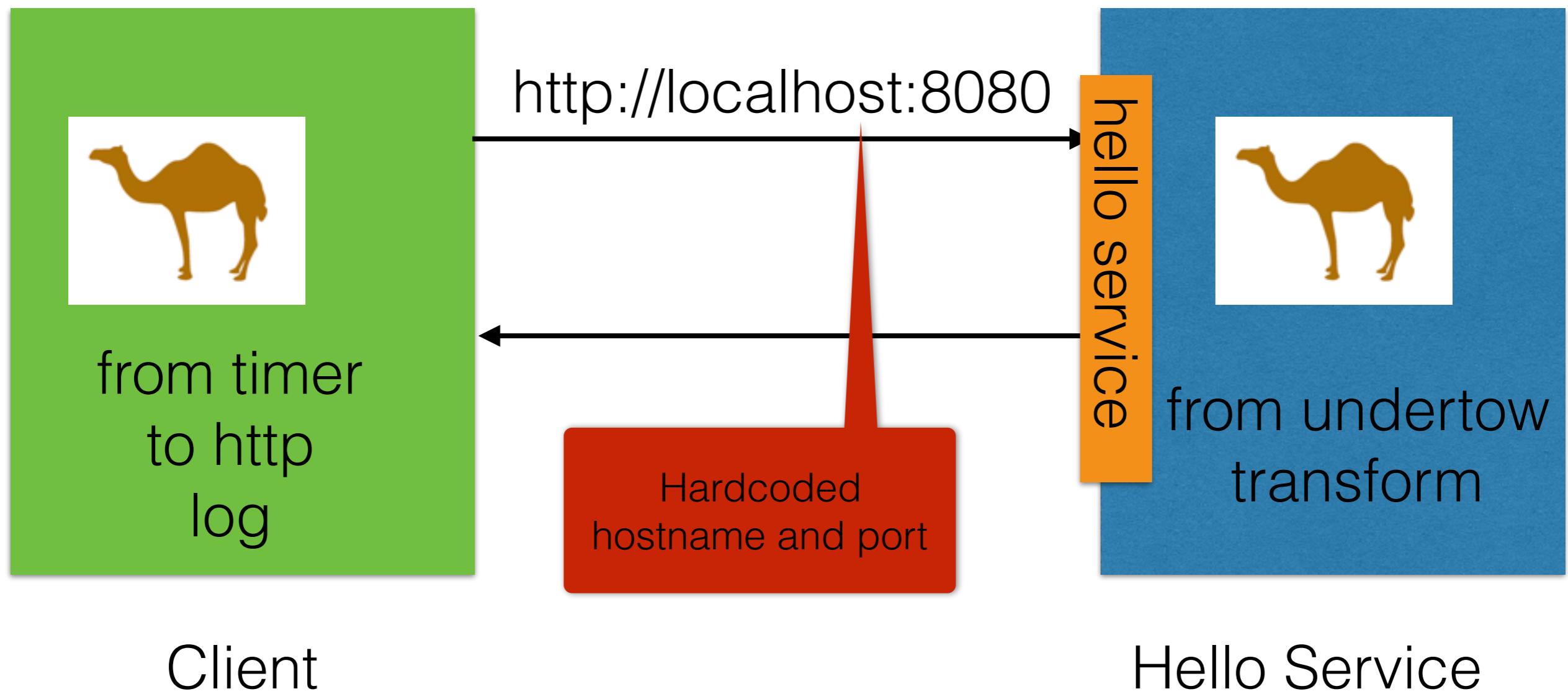
Hello Service

Client

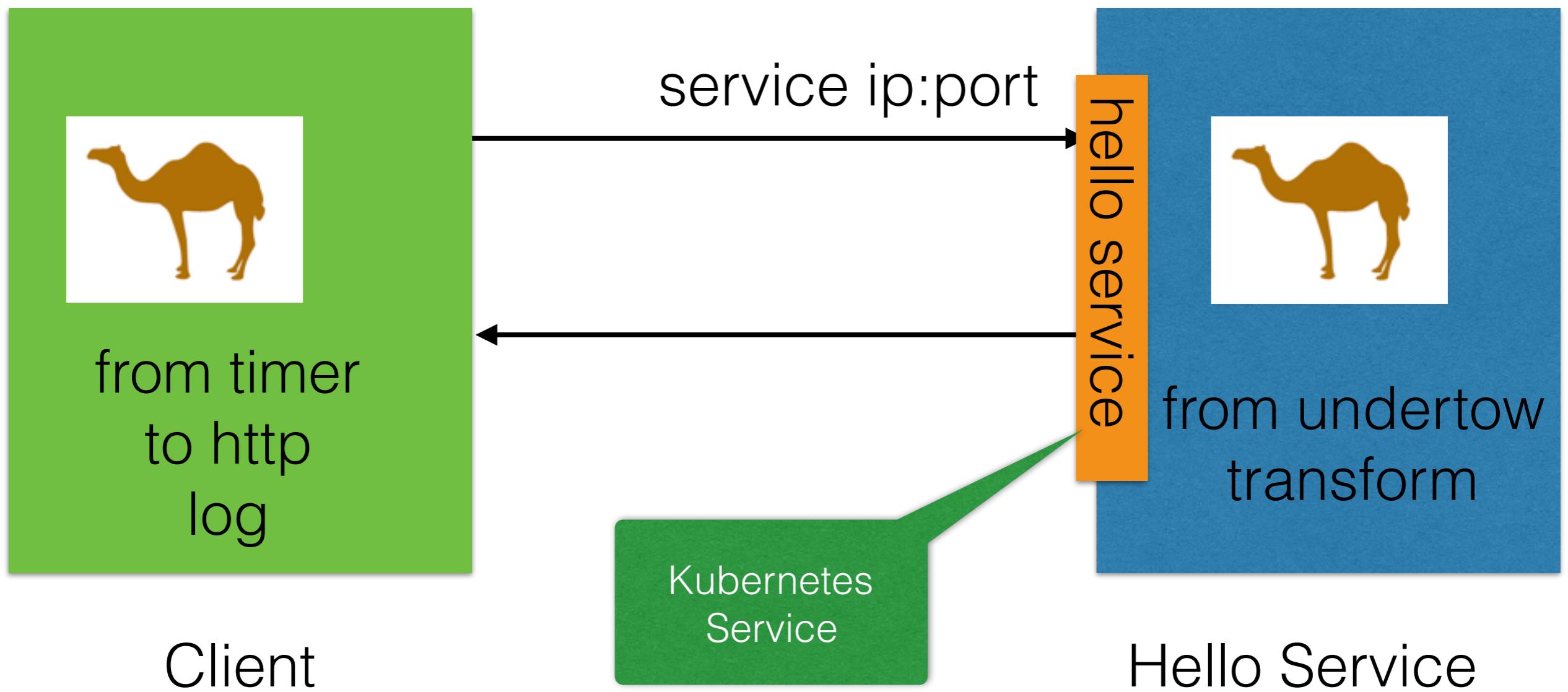
Our Demo



Static vs Dynamic Platform



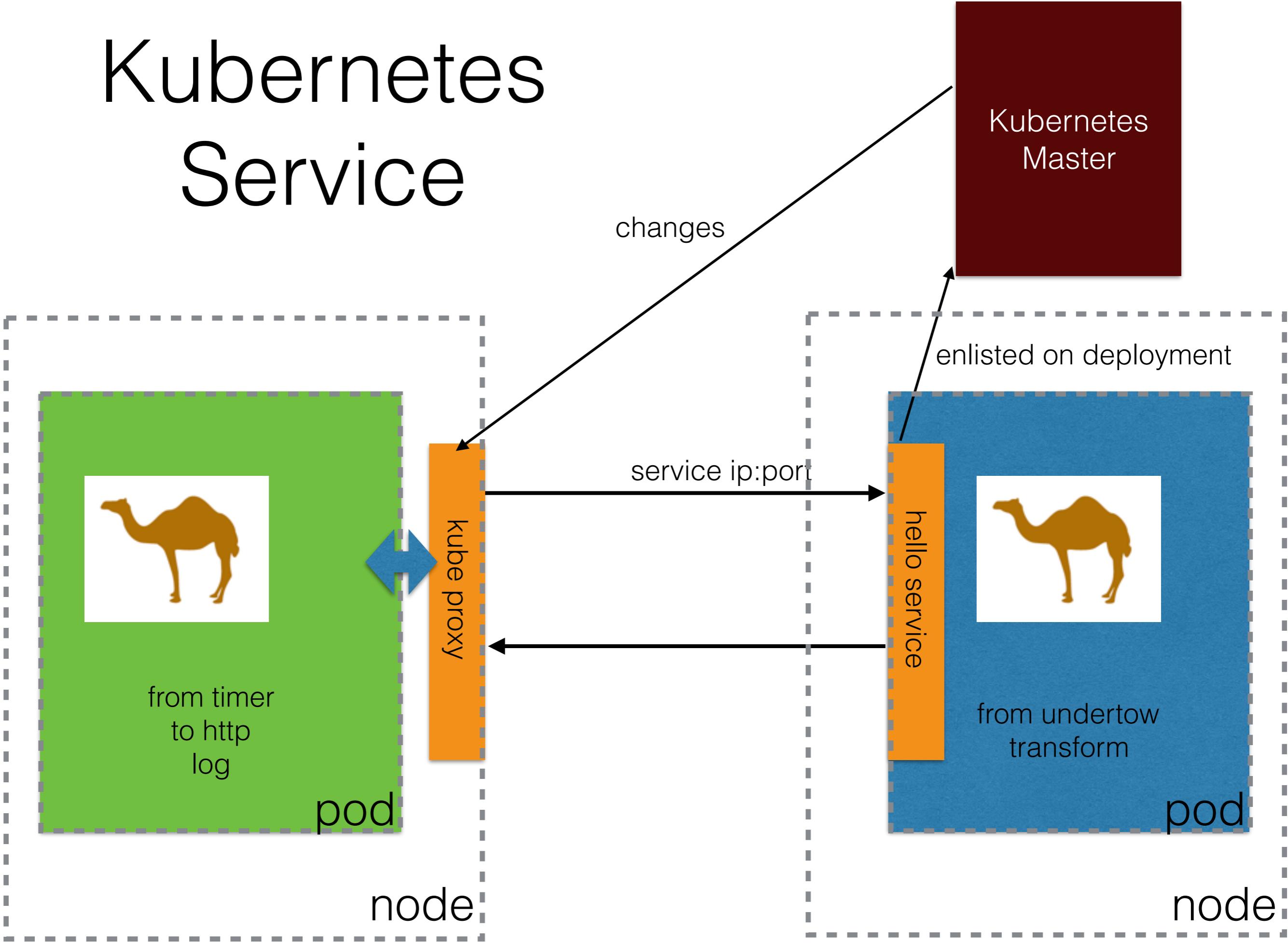
Dynamic Platform



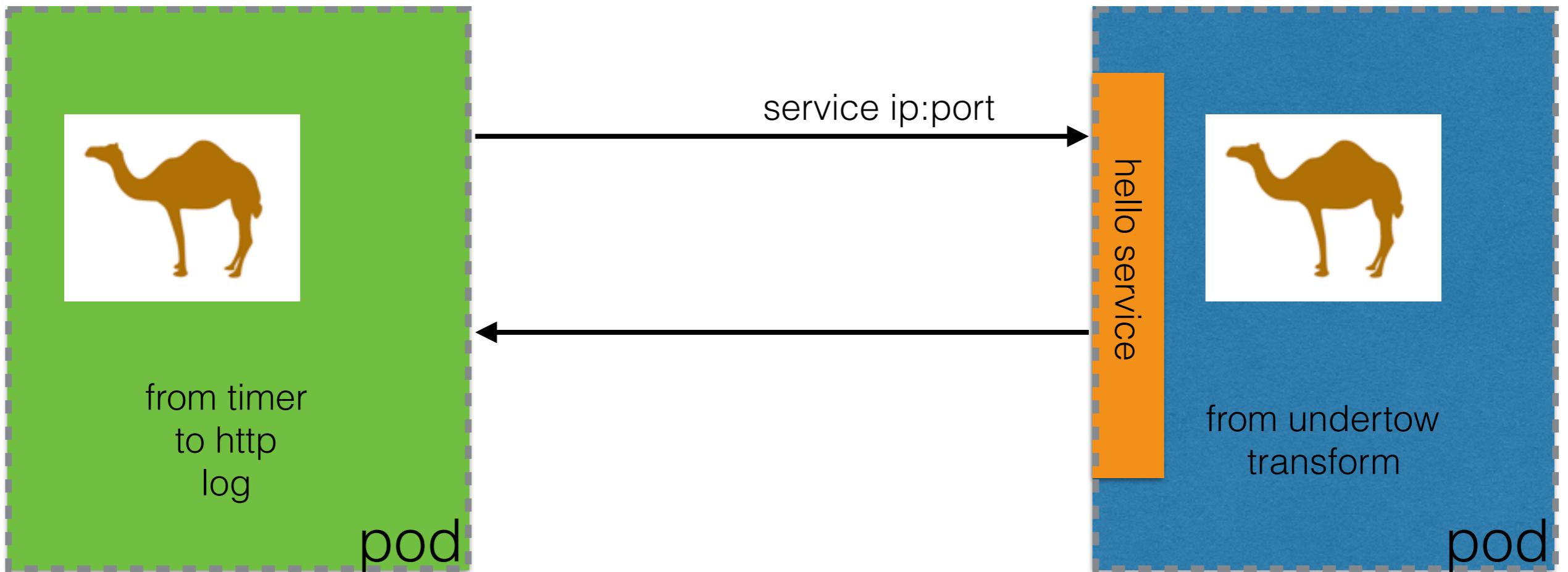
Kubernetes Service

- Network Connection to one or more Pods
- Unique static IP and port (lifetime of service)

Kubernetes Service



Kubernetes Service from user point of view



Out of the box service

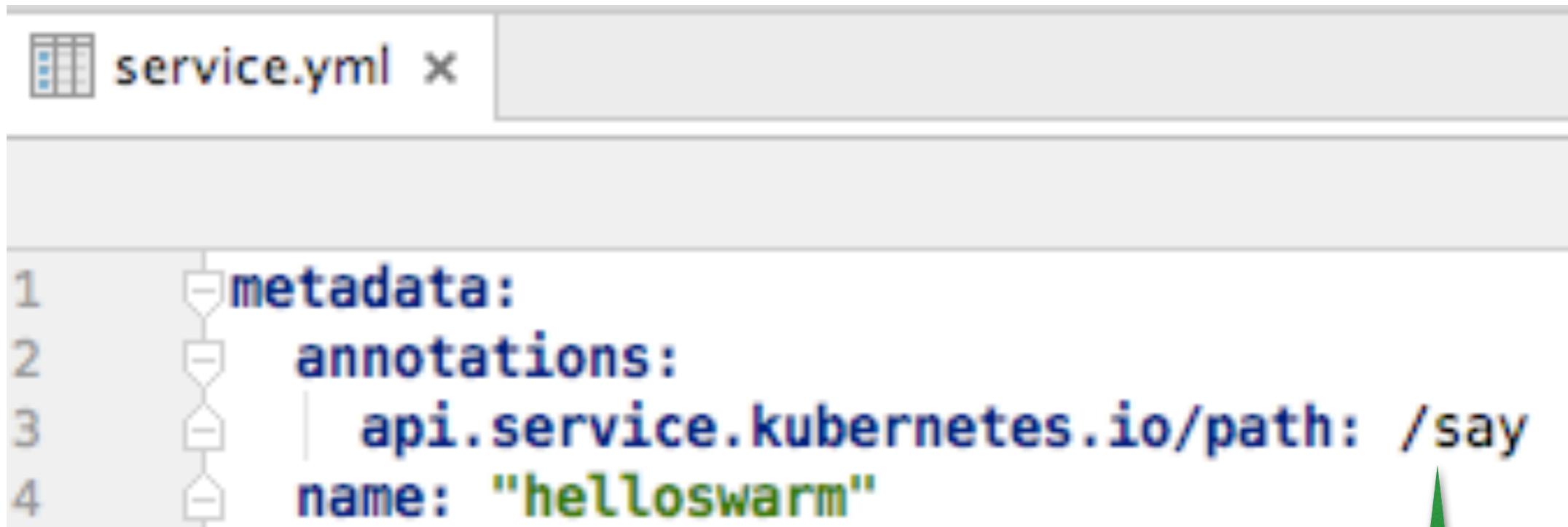
```
  - name: "helloswarm"
spec:
ports:
- name: "http"
  port: 80
  protocol: "TCP"
  targetPort: 8080
```

Maven artifactId

WildFly Swarm HTTP port

Service w/ Context Path

- Annotate Service with Context-Path



```
service.yaml
1 metadata:
2   annotations:
3     api.service.kubernetes.io/path: /say
4   name: "helloswarm"
```

src/main/fabric8/service.yaml

/say
is context-path

Custom Service

```
]}  
  metadata:  
    annotations:  
      |   api.service.kubernetes.io/path: "/say"  
      |   name: "myservice"  
    spec:  
      ports:  
        - name: "http"  
          port: 8181  
          protocol: "TCP"  
          targetPort: 8080  
          type: "LoadBalancer"  
src/main/fabric8/service.yml
```

The annotations section of the YAML file contains three key entries: `api.service.kubernetes.io/path: "/say"`, `name: "myservice"`, and `type: "LoadBalancer"`. Three green callout boxes with arrows point to these entries. The first arrow points to `name: "myservice"` and is labeled "Service Name". The second arrow points to `port: 8181` and is labeled "Service Port = Outside". The third arrow points to `targetPort: 8080` and is labeled "Container Port = Inside".

Using Kubernetes Service



from timer
to http
log

Client

We want to use hello service

How do we do that?

Using Kubernetes Service



from timer
to http
log

- Environment Variables

Client

- Hostname
- Port

```
export HELLOSWARM_SERVICE_HOST='172.30.196.15'  
export HELLOSWARM_SERVICE_PORT='80'
```

Service Discovery using DNS is available
in newer versions of Kubernetes.

Service using ENV



from timer
to http
log

- Use ENV

Client

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer:foo?period=2000")
            .to("netty4-http:http://{{service:helloswarm}}/say")
            .log("${body}");
    }
}
```

Service using DNS



from timer
to http
log

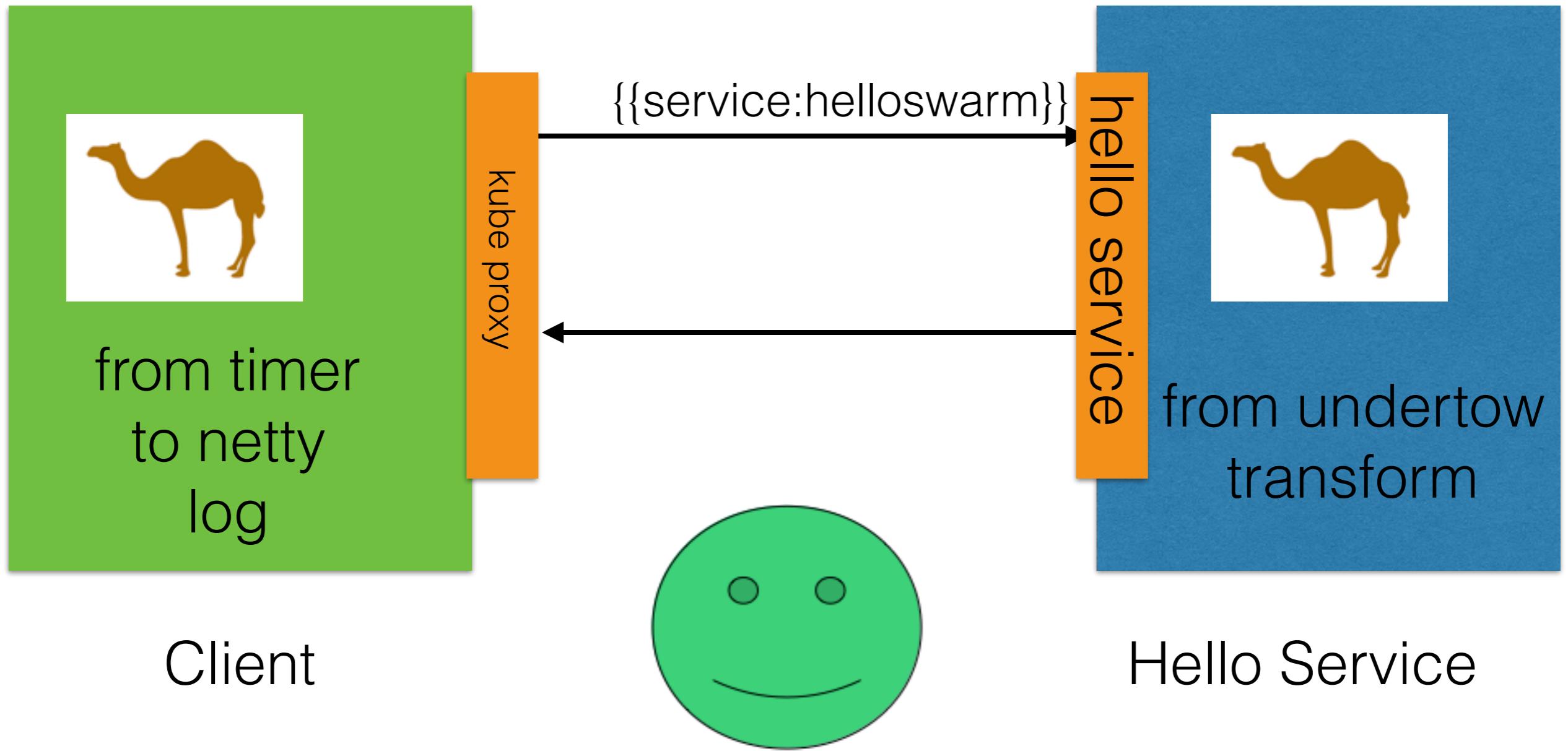
- Use DNS

Client

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer:foo?period=2000")
            .to("netty4-http:http://helloswarm/say")
            .log("${body}");
    }
}
```

Ready to run in Kubernetes



How to deploy to Kubernetes?

Maven Project



How to deploy to Kubernetes?



Deploy - Hello Service

hello service



from undertow
transform

Hello Service

- mvn fabric8:run
mvn fabric8:deploy

```
davsclaus:/Users/davsclaus/Documents/workspace/helloswarm/$ mvn fabric8:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Wildfly Swarm Example 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] >>> fabric8-maven-plugin:3.1.32:run (default-cli) > install @ hellosw
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ hellosw
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- fabric8-maven-plugin:3.1.32:resource (fmp) @ helloswarm ---
[INFO] F8> Running in Kubernetes mode
```

Deploy - Client



from timer
to http
log

- mvn fabric8:run
mvn fabric8:deploy

```
davsclaus:/Users/davsclaus/Documents/workspace/client/$ mvn fabric8:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building client 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] >>> fabric8-maven-plugin:3.1.32:run (default-cli) > install @ client >>>
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ client ---
```

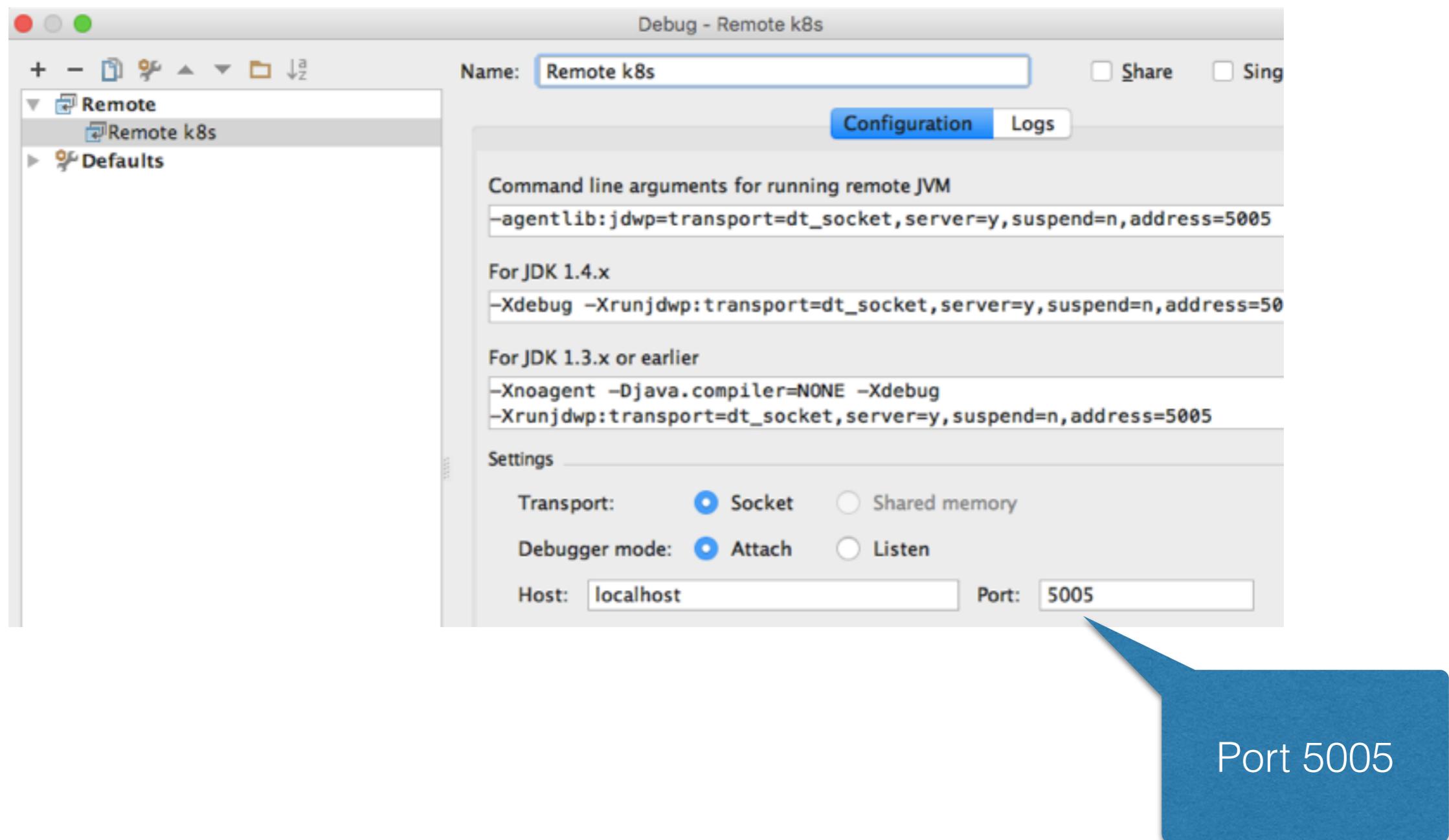
Debugging

- Debugging Pods

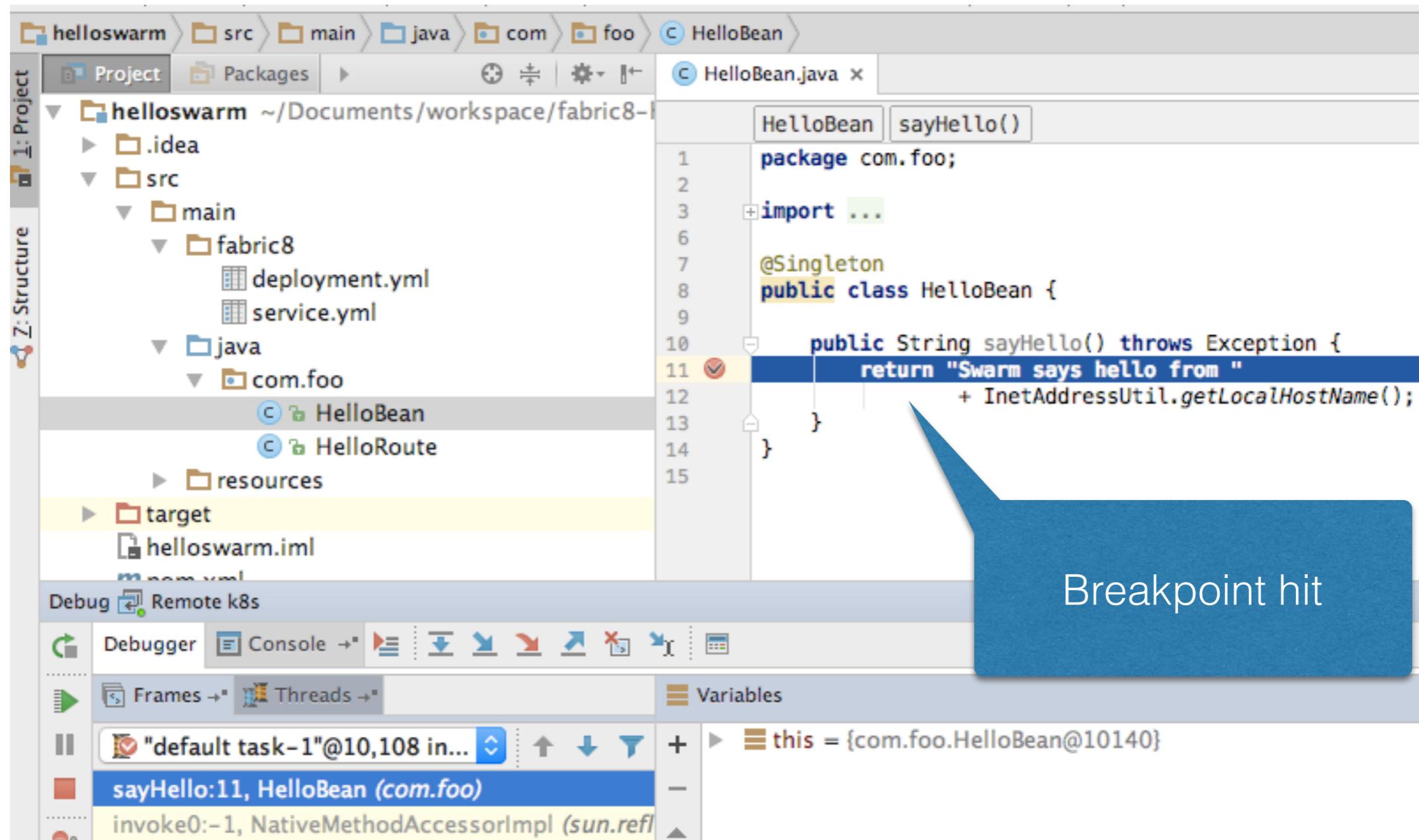
```
mvn fabric8:debug
```

```
[INFO] F8> Port forwarding to port 5005 on pod helloswarm-1942392107-13p2p using command: kubectl
[INFO] F8> Executing command: kubectl port-forward helloswarm-1942392107-13p2p 5005:5005
[INFO] F8>
[INFO] F8> Now you can start a Remote debug execution in your IDE by using localhost and the debug port 5005
[INFO] F8>
[INFO] kubectl> Forwarding from 127.0.0.1:5005 -> 5005
[INFO] kubectl> Forwarding from [::1]:5005 -> 5005
[INFO] kubectl> Handling connection for 5005
```

Remote Debug



Remote Debug



Running Local Kubernetes/OpenShift

- Fabric8 Maven Plugin
- MiniKube
- MiniShift
- Vagrant
- OpenShift CDK

<https://fabric8.io/guide/getStarted/index.html>

Running fabric8

- Download gofabric8

<https://github.com/fabric8io/gofabric8/releases>

- Start fabric8

gofabric8 start --minishift --console --memory=2000

<https://fabric8.io/guide/getStarted/minishift.html>

Running fabric8

```
mvn  
io.fabric8:  
fabric8-maven-plugin:3.1.63:  
cluster-start  
-Dfabric8.cluster.kind=openshift
```

How I installed OpenShift

```
davsclaus:/Users/davsclaus/Documents/workspace/$ mvn io.fabric8:fabric8-maven-plugin:3.1.63:cluster-start -Dfabric8.cluster.kind=openshift
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----
[INFO]
[INFO] --- fabric8-maven-plugin:3.1.63:cluster-start (default-cli) @ standalone-pom ---
[INFO] Found gofabric8 at: /Users/davsclaus/.fabric8/bin/gofabric8
[INFO] F8> Running command gofabric8 /Users/davsclaus/.fabric8/bin/gofabric8 version --batch
[INFO] gofabric8> gofabric8, version 0.4.76 (branch: 'master', revision: 'cc713a8')
[INFO] gofabric8> build date:      '20161002-01:58:36'
[INFO] gofabric8> go version:     '1.7.1'
[INFO] F8> running: gofabric8 start --batch --minishift --console
[INFO] F8> Running command gofabric8 /Users/davsclaus/.fabric8/bin/gofabric8 start --batch --minishift --co
nsole
[INFO] gofabric8> fabric8 recommends OSX users use the xhyve driver
[INFO] gofabric8> xhyve driver already installed
```

Installing is easy
(even from a bar with a drink)



<https://vimeo.com/185299147>

fabric8 Web Console

The screenshot shows the fabric8 Web Console interface. The top navigation bar includes back, forward, and search icons, followed by the URL `192.168.64.5:31511/workspaces/default/namespace/default/services?q=`. Below the URL is a dark header bar with tabs for **fabric8**, **Teams**, **default**, **Runtime**, and **Services**. The **Services** tab is active. On the left, a sidebar menu lists **Overview**, **Services** (which is selected and highlighted in blue), **Deployments**, **Replicas**, and **Pods**. The main content area features a search bar labeled "Filter services..." with an "x" button. Below the search bar is a table listing five services:

<input type="checkbox"/>	Name	Address	Pods
<input type="checkbox"/>	client	172.46.103.162	
<input type="checkbox"/>	fabric8	172.46.226.23	
<input type="checkbox"/>	helloswarm	172.46.101.229	
<input type="checkbox"/>	kubernetes	172.30.0.1:443 172.30.0.1:53 172.30.0.1:53	

minishift service fabric8

OpenShift CLI

You can also use CLI from
docker &
kubernetes

- oc get pods

```
davsclaus:/Users/davsclaus/Documents/workspace/client/$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
client-1-60c00         1/1     Running   5          9m
exposecontroller-1-2fb1p 1/1     Running   0          35m
fabric8-cg8c0          1/1     Running   0          36m
helloswarm-1-qcnv7    1/1     Running   0          14m
```

OpenShift CLI

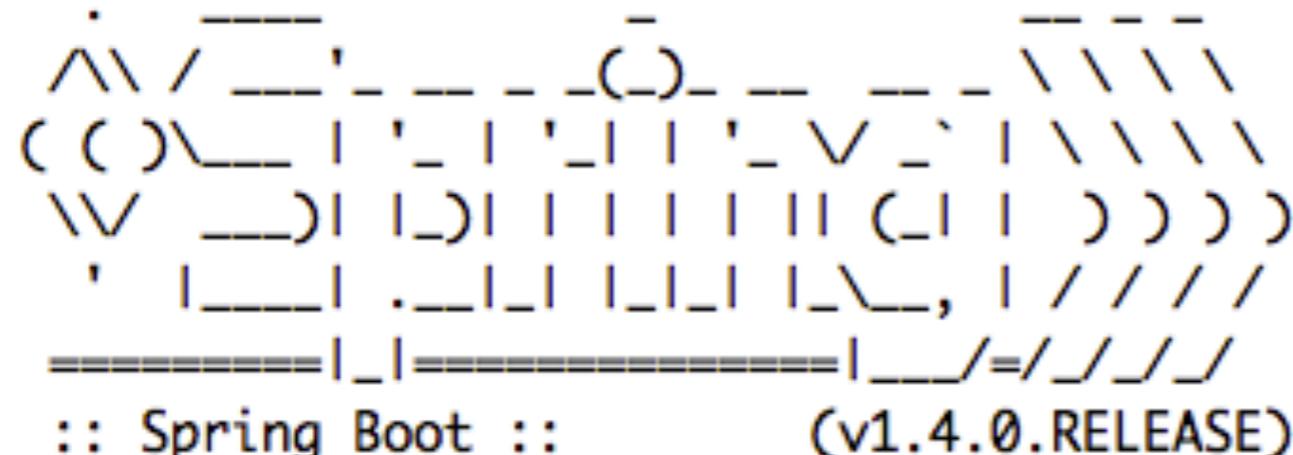
- oc get service

```
davsclaus:/Users/davsclaus/Documents/workspace/$ oc get service
NAME           CLUSTER-IP      EXTERNAL-IP        PORT(S)
client         172.30.14.254   172.46.103.162,172.46.103.162   8080/TCP
fabric8        172.30.140.79   172.46.226.23,172.46.226.23   80/TCP
hello          172.30.21.39    172.46.203.233,172.46.203.233  8181/TCP
kubernetes     172.30.0.1       <none>              443/TCP,5
```

OpenShift CLI

- `oc logs -f <pod name>`

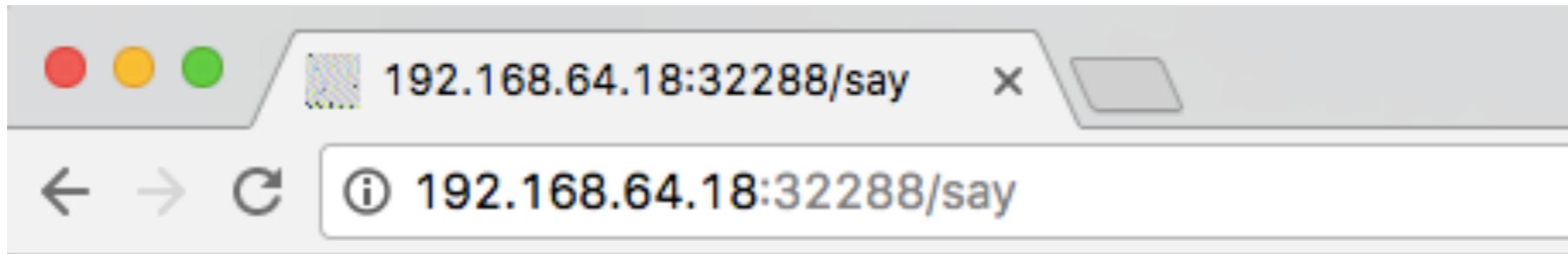
```
davsclaus:/Users/davsclaus/Documents/workspace/$ oc logs -f client-1-qlfym
I> No access restrictor found, access to any MBean is allowed
Jolokia: Agent started with URL http://172.17.0.3:8778/jolokia/
2016-09-09 13:07:33.604:INFO:ifasjipjsoejs.Server:jetty-8.y.z-SNAPSHOT
2016-09-09 13:07:33.649:INFO:ifasjipjsoejs.AbstractConnector:Started Select
```



Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7

Access Service from your laptop

- minishift service helloswarm



Swarm says hello from helloswarm-6-m3zi2

Scaling

- Change deployment replicas

	Name	▲	Pods	Scale
	camel client		1	1
	image exposecontroller		1	1
	camel helloswarm		1	1

	Name	▲	Pods	Scale
	camel client		1	1
	image exposecontroller		1	1
	camel helloswarm		2	2

Load balancing
is random

Scaling

- Service load balancing

```
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-665c1
```

Error Handling

- Client Side Redelivery

```
@Override  
public void configure() throws Exception {  
    onException(Exception.class)  
        .maximumRedeliveries(10)  
        .redeliveryDelay(2000);  
  
    from("timer:foo?period=2000")  
        .to("netty4-http:http:{service:helloswarm}/say"  
        .log("${body}");  
}
```

Error Handling

- Client Side Circuit Breaker

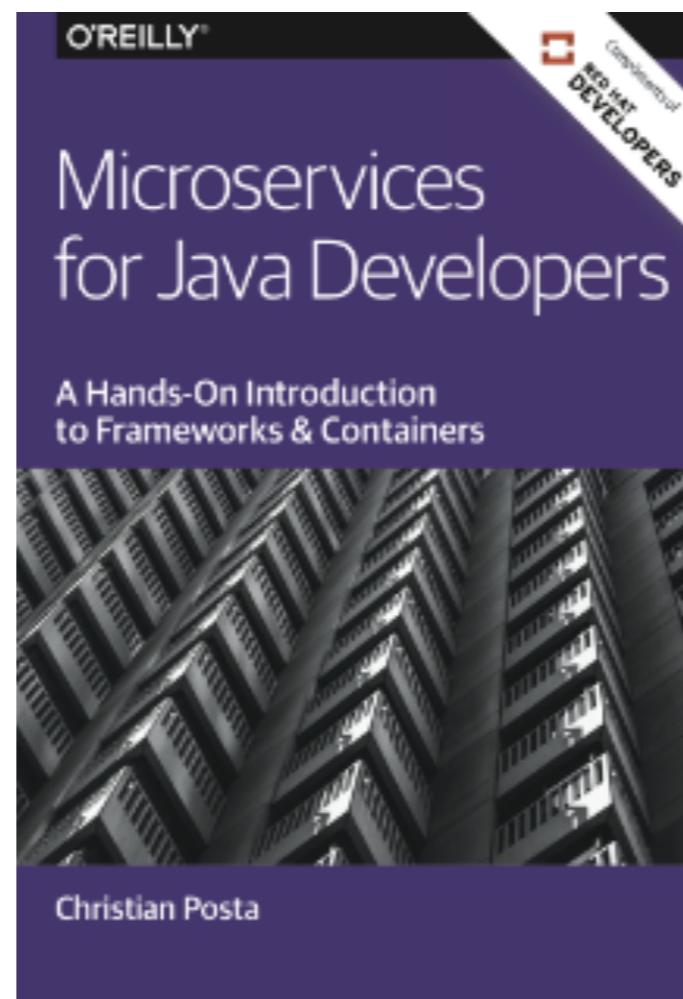
```
@Override  
public void configure() throws Exception {  
    from("timer:foo?period=2000")  
        .hystrix()  
            .to("netty4-http:http://{{service:helloswarm}}/say"  
        .onFallback()  
            .setBody().constant("Nobody want to talk to me")  
        .end()  
            .log("${body}");  
}
```

Angry Pods



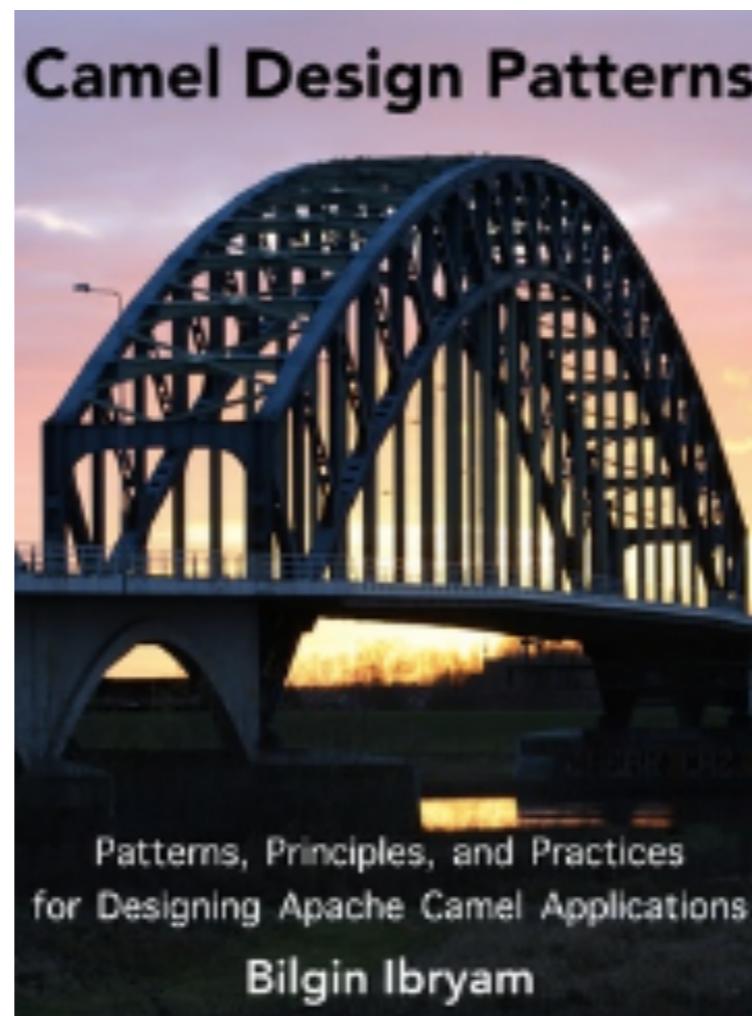
1st person shooter - Kill your pods

Free Book



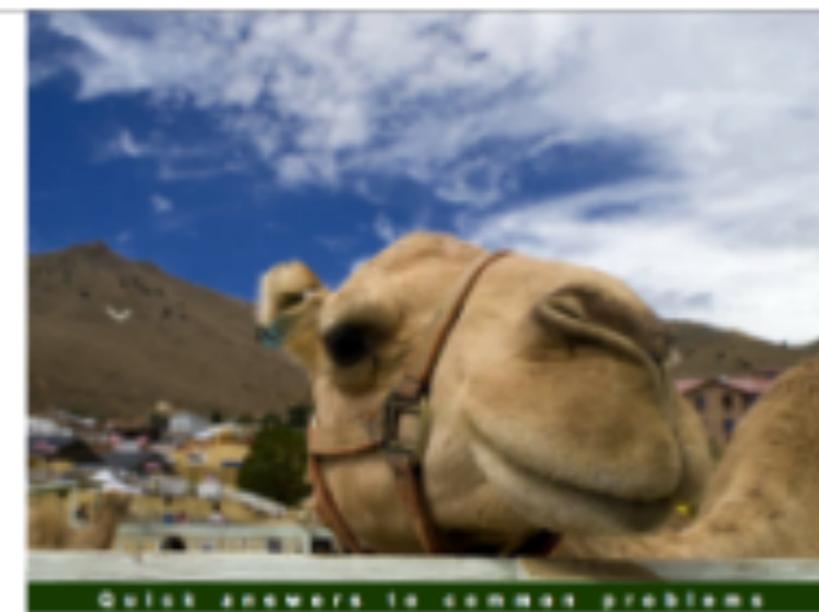
[http://developers.redhat.com/promotions/
microservices-for-java-developers/](http://developers.redhat.com/promotions/microservices-for-java-developers/)

Non Free Book



<https://leanpub.com/camel-design-patterns>

Non Free Book



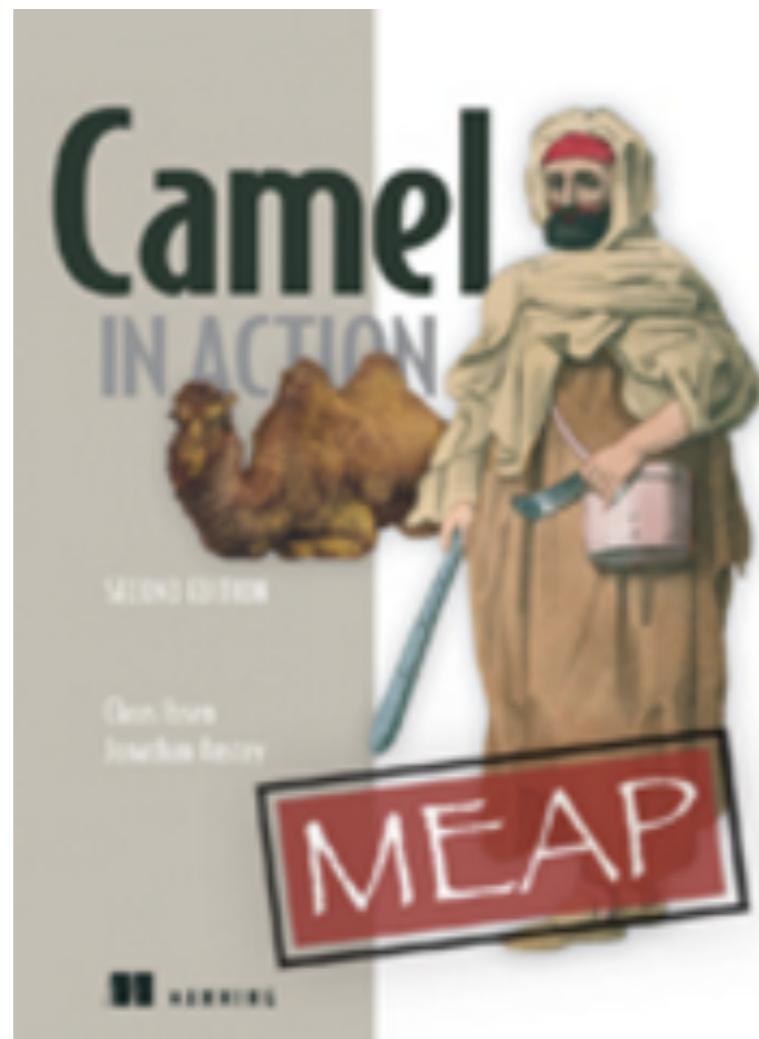
Apache Camel Developer's Cookbook

Solve common integration tasks with over 100 easily accessible
Apache Camel recipes

Scott Cranton Jakub Konečný [PACKT] enterprise™

[https://www.packtpub.com/application-development/
apache-camel-developers-cookbook](https://www.packtpub.com/application-development/apache-camel-developers-cookbook)

Non Free Book



Coupon code: camel39
gives 39% discount

<http://manning.com/ibsen2>

Links



- fabric8
 - <http://fabric8.io>
 - Demo source code
 - <https://github.com/davsclaus/fabric8-hello>
 - Try fabric8
 - <https://fabric8.io/guide/getStarted/minishift.html>
 - Videos, blogs and more
 - <https://fabric8.io/community/index.html>



@davsclaus



davsclaus

davsclaus.com