

Developing Cloud Ready Camel Microservices

May 2nd 2017

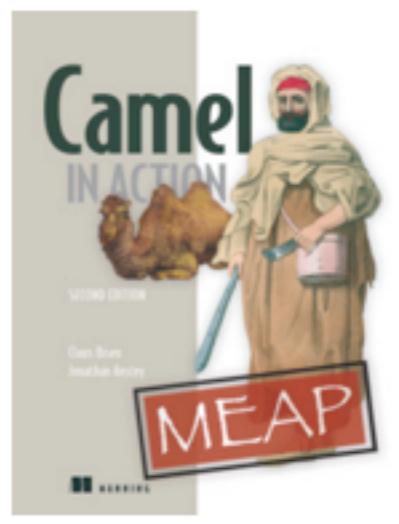
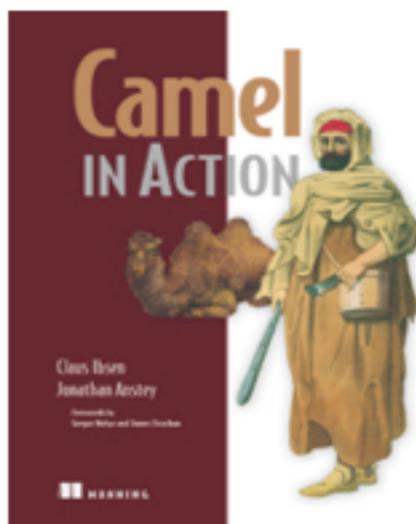
Claus Ibsen



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

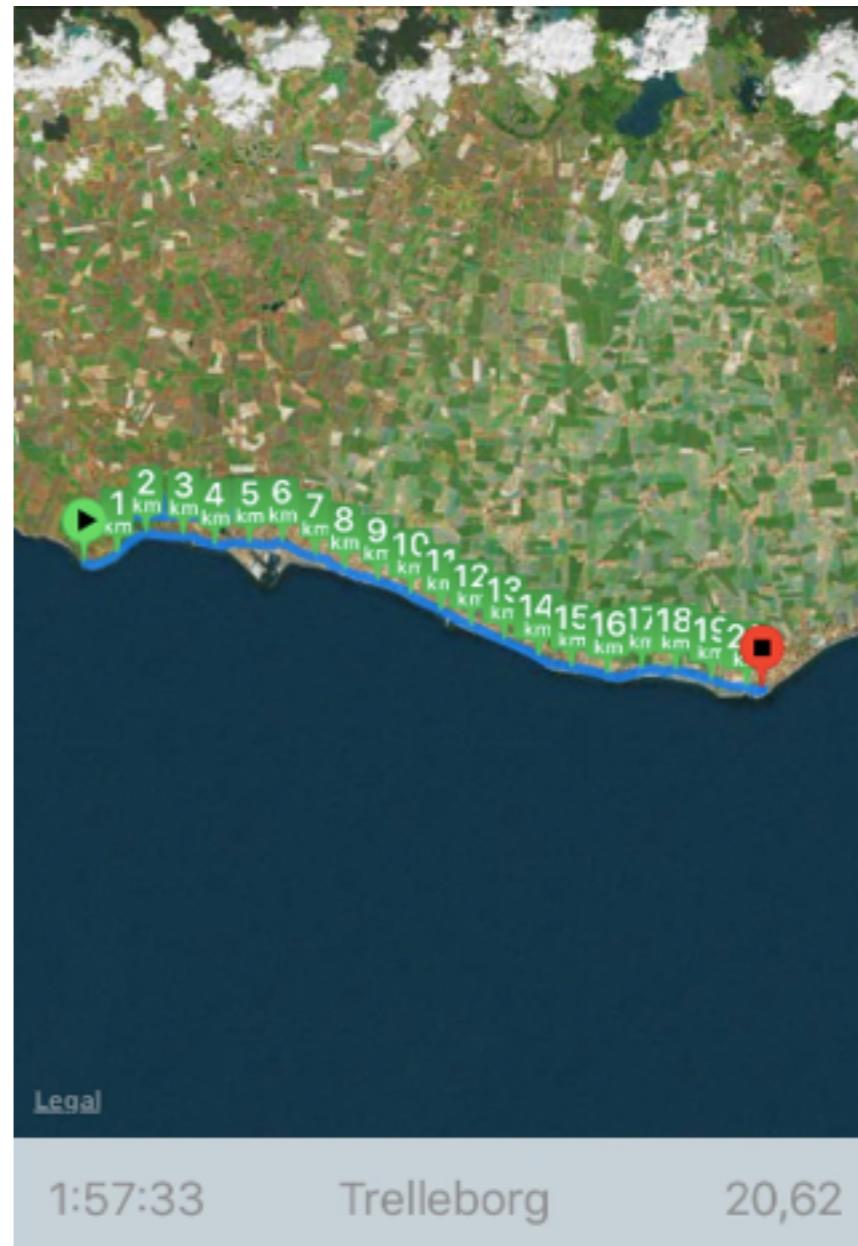


@davsclaus
davsclaus
davsclaus.com

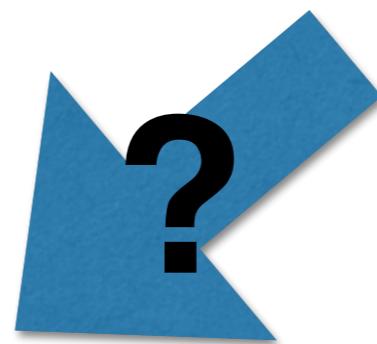
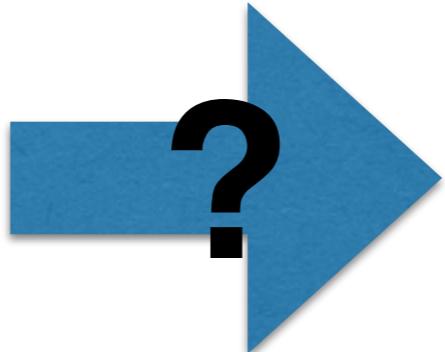


Senior Developer vs Real Life

- Completed my first half marathon 3 days ago



Key Message



or



Tip of Iceberg



Running Local OpenShift / Kubernetes

- MiniShift
- OpenShift CDK
- MiniKube (Kubernetes)



Running OpenShift Locally

- Download MiniShift

<https://github.com/minishift/minishift/releases>

- Start MiniShift

minishift start --openshift-version v1.5.0

minishift config set openshift-version 1.5.0

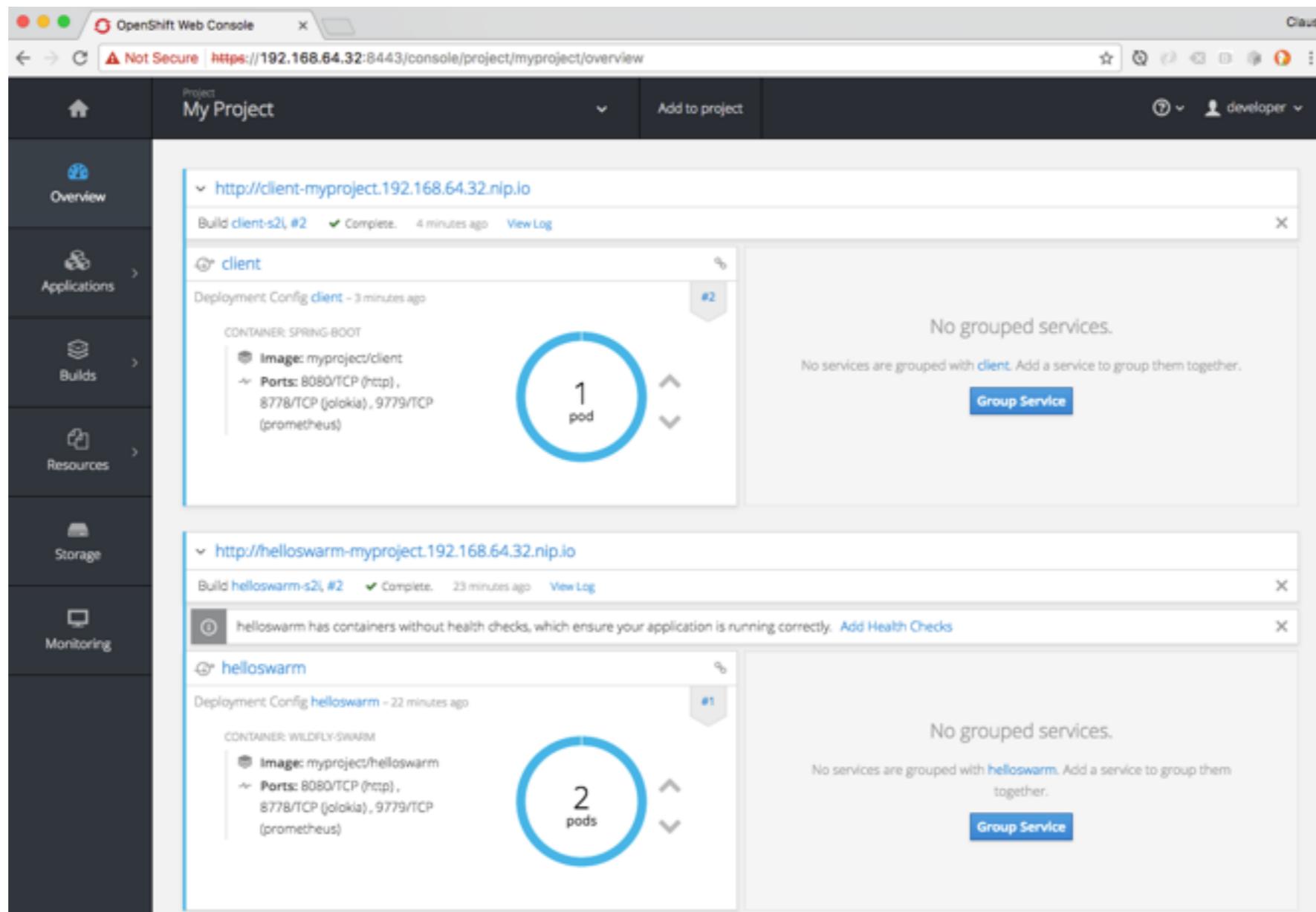
TIP To force a specific version

<https://www.openshift.org/minishift/>

How I installed OpenShift locally

```
davsclaus:/Users/davsclaus/$ minishift start --openshift-version v1.5.0
Starting local OpenShift cluster using 'xhyve' hypervisor...
Downloading ISO 'https://github.com/minishift/minishift-b2d-iso/releases/download/v1.0.2/minishift-b2d.iso'
 40.00 MB / 40.00 MB [=====]
Downloading OpenShift binary 'oc' version 'v1.5.0'
 18.93 MB / 18.93 MB [=====
-- Checking OpenShift client ... OK
-- Checking Docker client ... OK
-- Checking Docker version ... OK
-- Checking for existing OpenShift container ... OK
-- Checking for openshift/origin:v1.5.0 image ...
  Pulling image openshift/origin:v1.5.0
    Pulled 0/3 layers, 3% complete
    Pulled 1/3 layers, 72% complete
    Pulled 2/3 layers, 99% complete
    Pulled 3/3 layers, 100% complete
  Extracting
  Image pull complete
-- Checking Docker daemon configuration ... OK
-- Checking for available ports ... OK
-- Checking type of volume mount ...
  Using Docker shared volumes for OpenShift volumes
-- Creating host directories ... OK
```

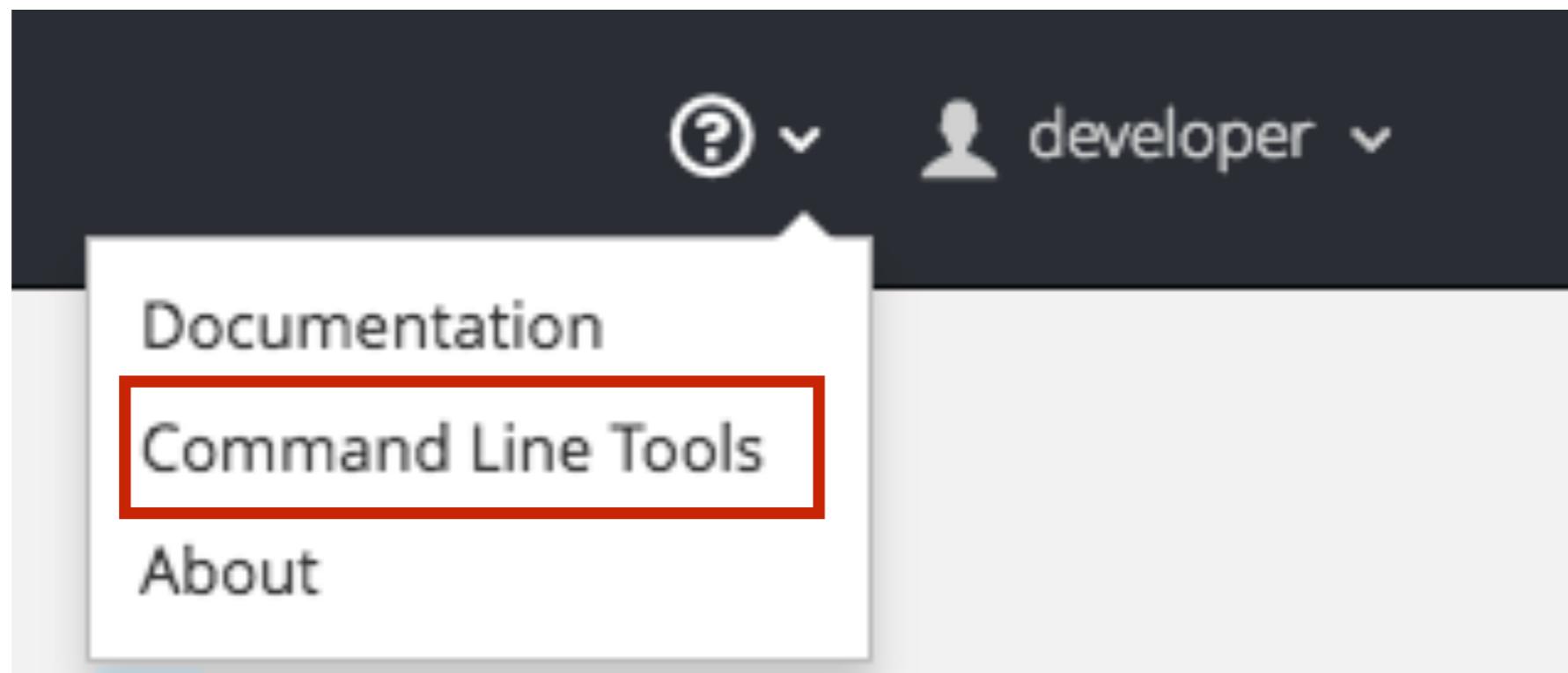
OpenShift Web Console



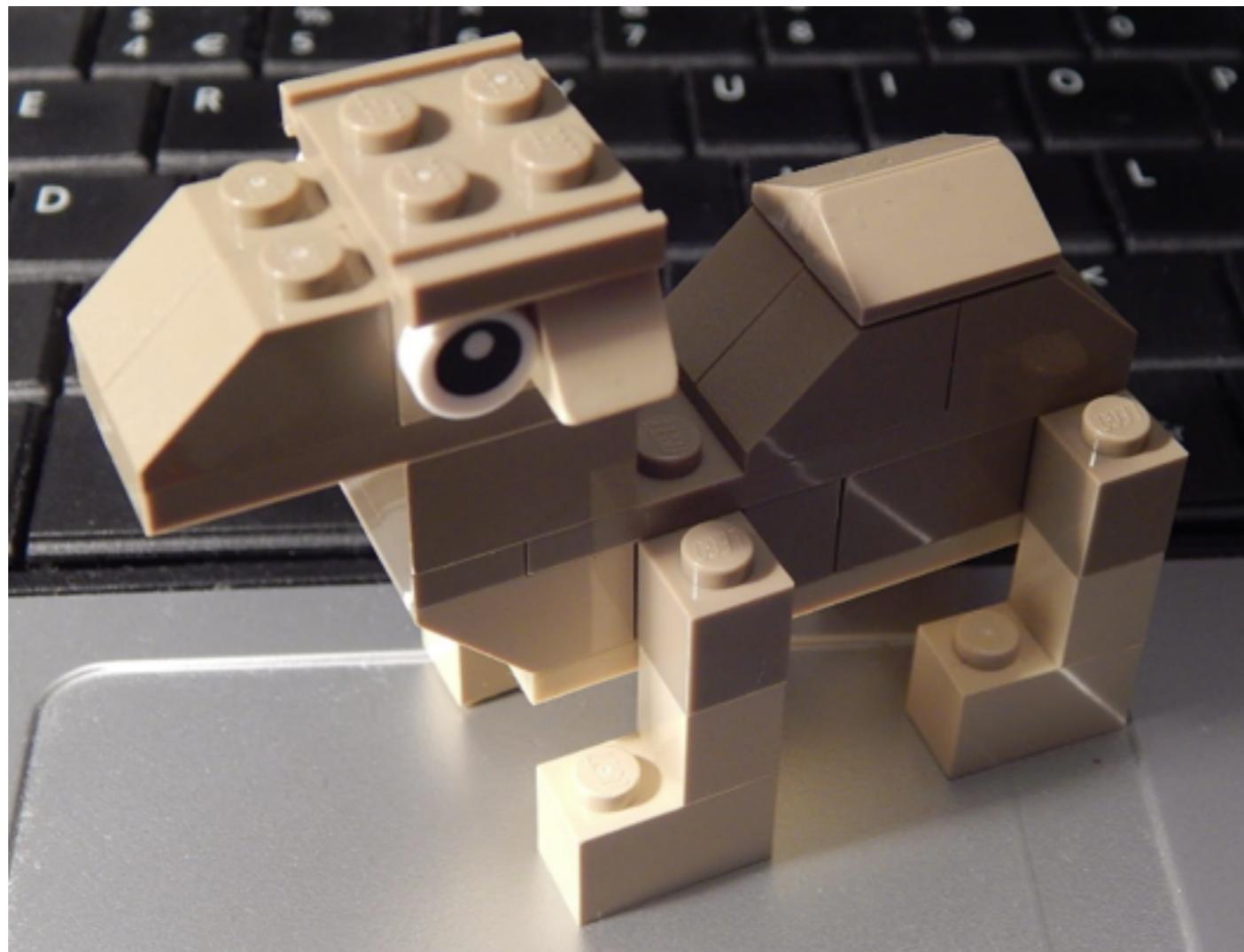
minishift console

OpenShift CLI (oc) Installation

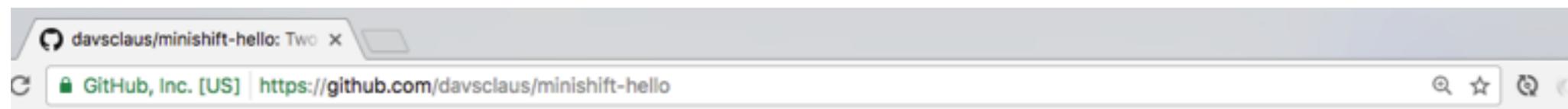
- Instructions from Web Console



Build a Camel Demo Time



Source Code & Slides

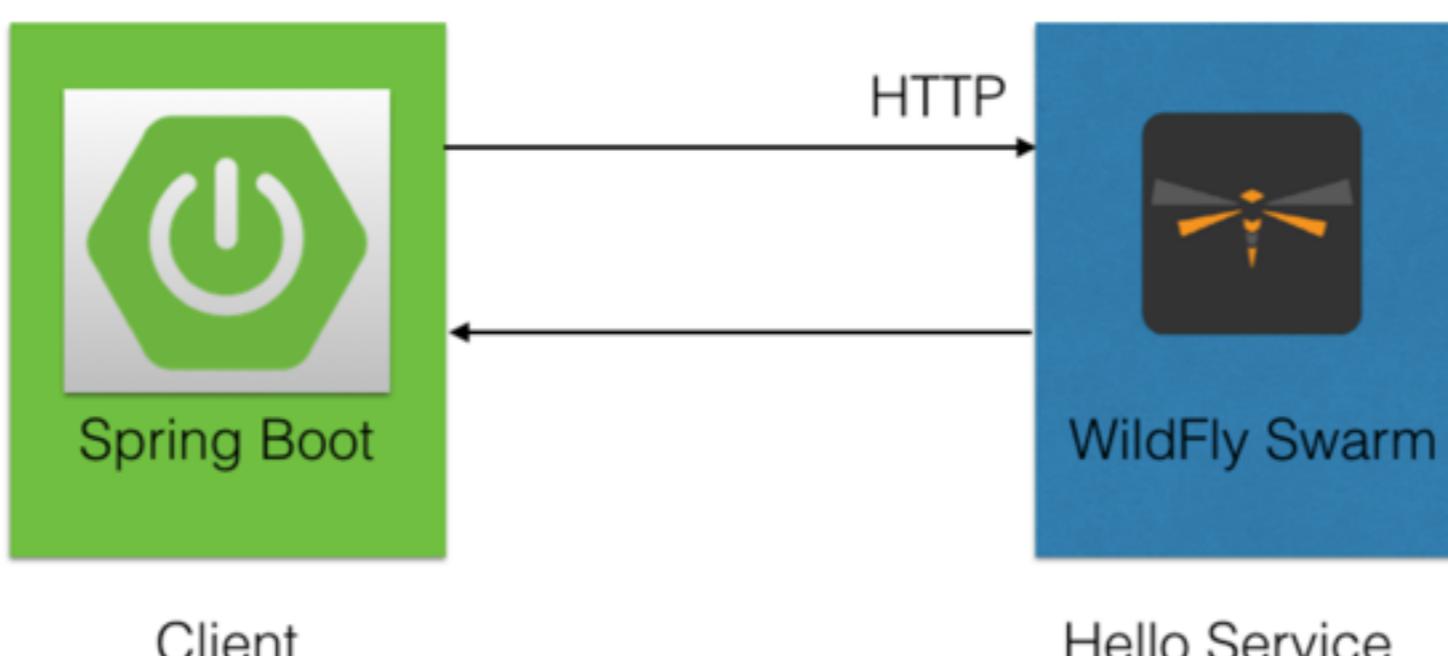
A screenshot of a web browser window. The title bar says "davsclaus/minishift-hello: Two". The address bar shows "GitHub, Inc. [US] https://github.com/davsclaus/minishift-hello". The main content area is titled "minishift-hello" and contains the following text:

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

There are three Maven projects:

- client - Spring Boot application with Camel that triggers every 2nd second to call the hello service and log the response. The client uses Camel client side retry for error handling.
- client-hystrix - A client that uses Hystrix as circuit breaker for error handling.
- helloswarm - WildFly Swarm application hostin a hello service which returns a reply message.

The diagram below illustrates this:



```
graph LR; Client[Spring Boot Client] -- HTTP --> HelloService[Hello Service]; HelloService -- HTTP --> Client;
```

The diagram shows two boxes representing microservices. The left box is green and labeled "Spring Boot Client" at the bottom, containing a white power button icon inside a hexagon. The right box is blue and labeled "Hello Service" at the bottom, containing a yellow and orange stylized insect icon. Two horizontal arrows connect them: one pointing from the client to the service, labeled "HTTP" above it, and another pointing from the service back to the client, also labeled "HTTP" above it.

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

Hello Service

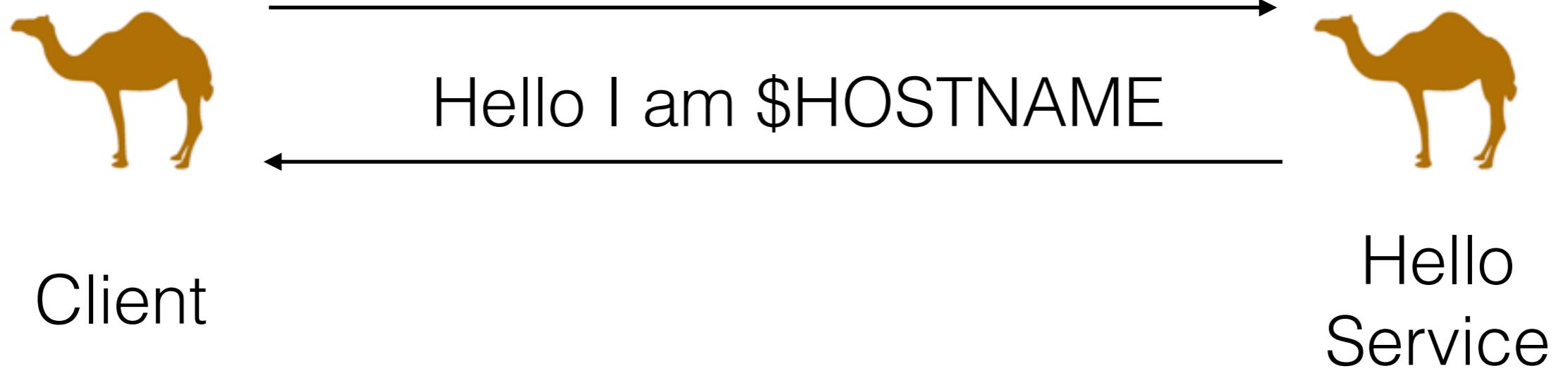


Client

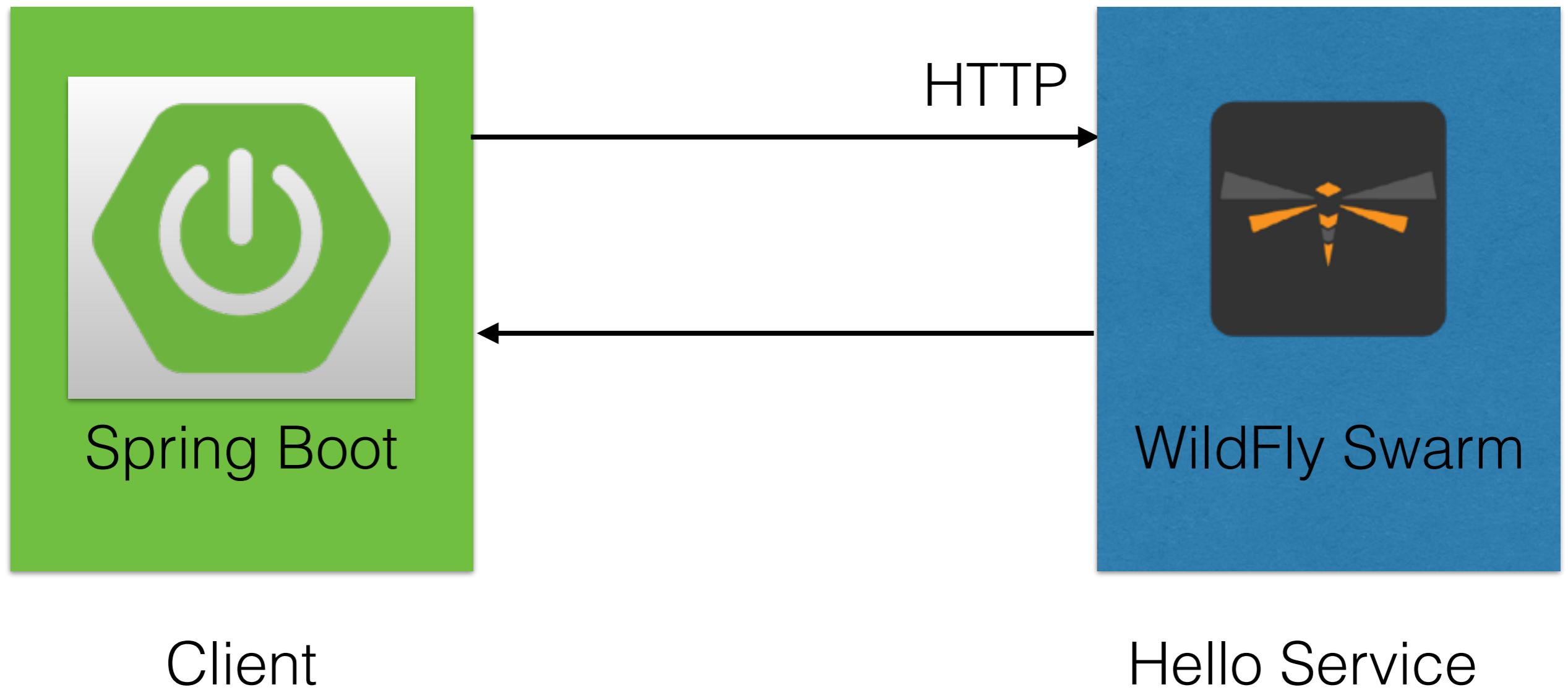


Hello
Service

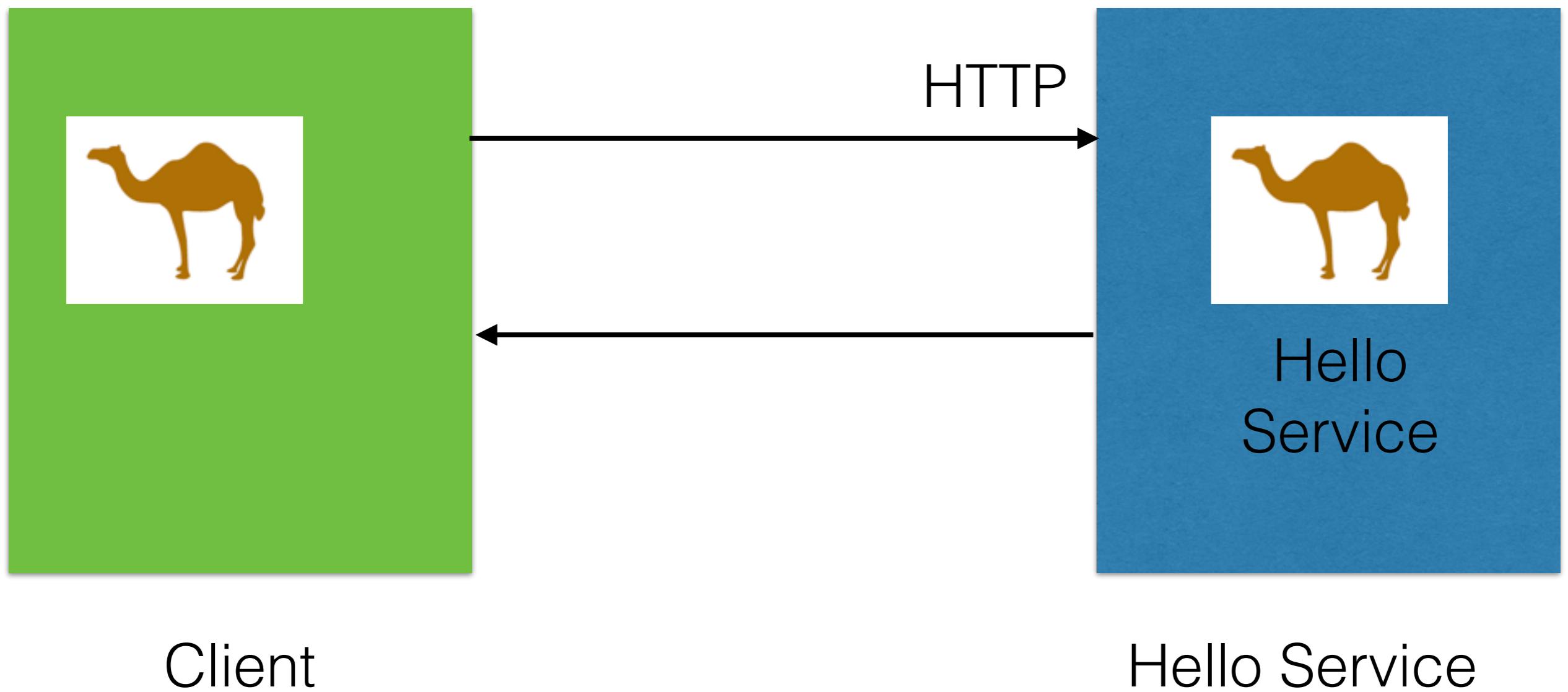
Hello Service



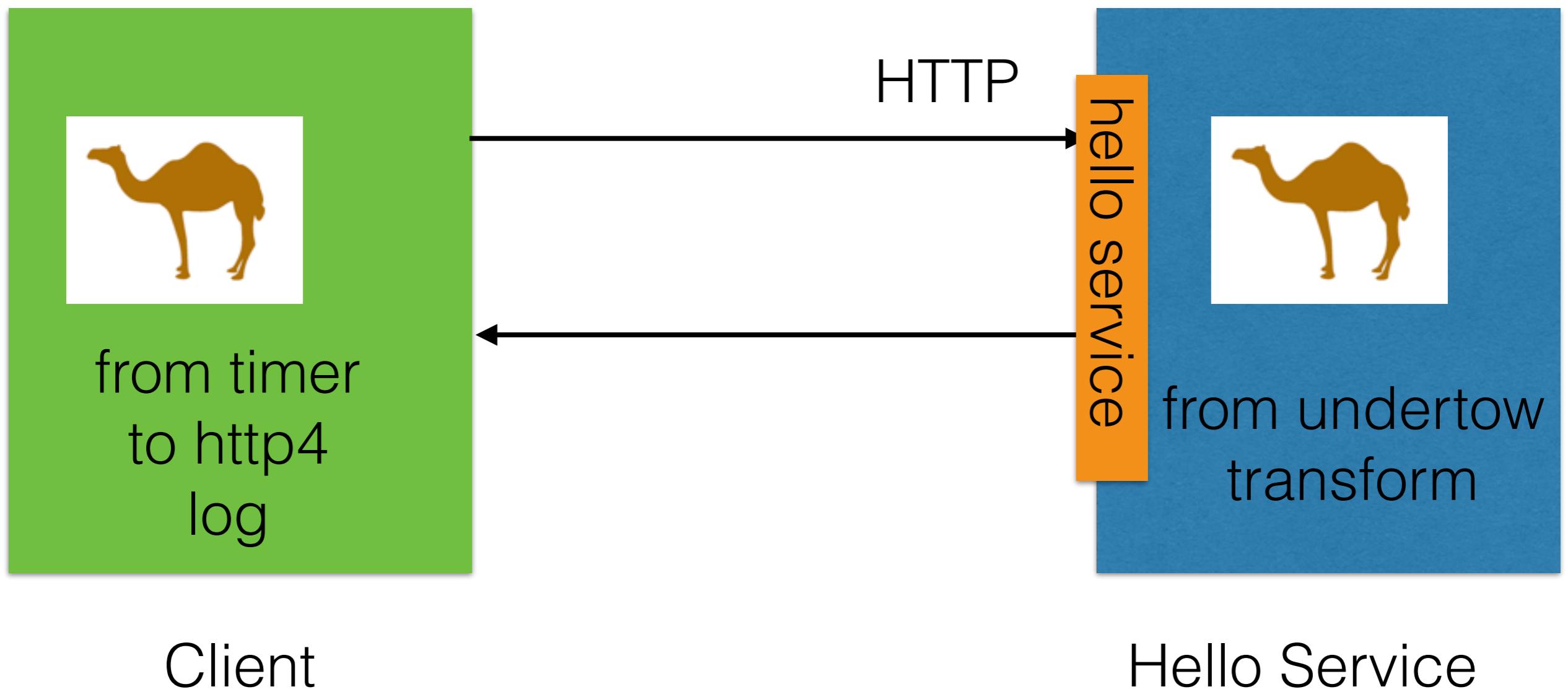
Implementation



Implementation



Implementation



WildFly Swarm Generator



WildFly Swarm Project General

wildfly-swarm.io/generator/

WildFly Swarm Project Generator

Rightsize your Java EE microservice in a few clicks

Instructions

1. Choose the dependencies you need
2. Click on the Generate button to download the *helloswarm.zip* file
3. Unzip the file in a directory of your choice
4. 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 filtered by :

Selected dependencies



Hello Service

```
@Singleton
public class HelloRoute extends RouteBuilder {

    @Inject
    @Uri("undertow:http://0.0.0.0:8080/hello")
    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 {  
        String answer = "Swarm says hello from "  
        |   + InetAddressUtil.getLocalHostName();  
        return answer;  
    }  
}
```



Spring Boot Starter

A screenshot of a web browser window titled "Spring Initializr". The URL bar shows "start.spring.io". The main heading is "SPRING INITIALIZR" followed by the subtext "bootstrap your application now".

Generate a Maven Project with Spring Boot 1.5.3

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 X

Actuator X

Apache Camel X

Generate Project ⌘ + ↵



Client

```
@Component  
public class MyRoute extends RouteBuilder {  
  
    @Override  
    public void configure() throws Exception {  
        from( uri: "timer:foo?period=2000")  
            .to("http4:localhost:8080/hello")  
            .log("${body}");  
    }  
}
```

Great plugin
for IDEA users

Apache Camel IDEA plugin



The screenshot shows a Java code editor with the following code:

```
@Override
public void configure() throws Exception {
    from("timer:too?").delay
}
```

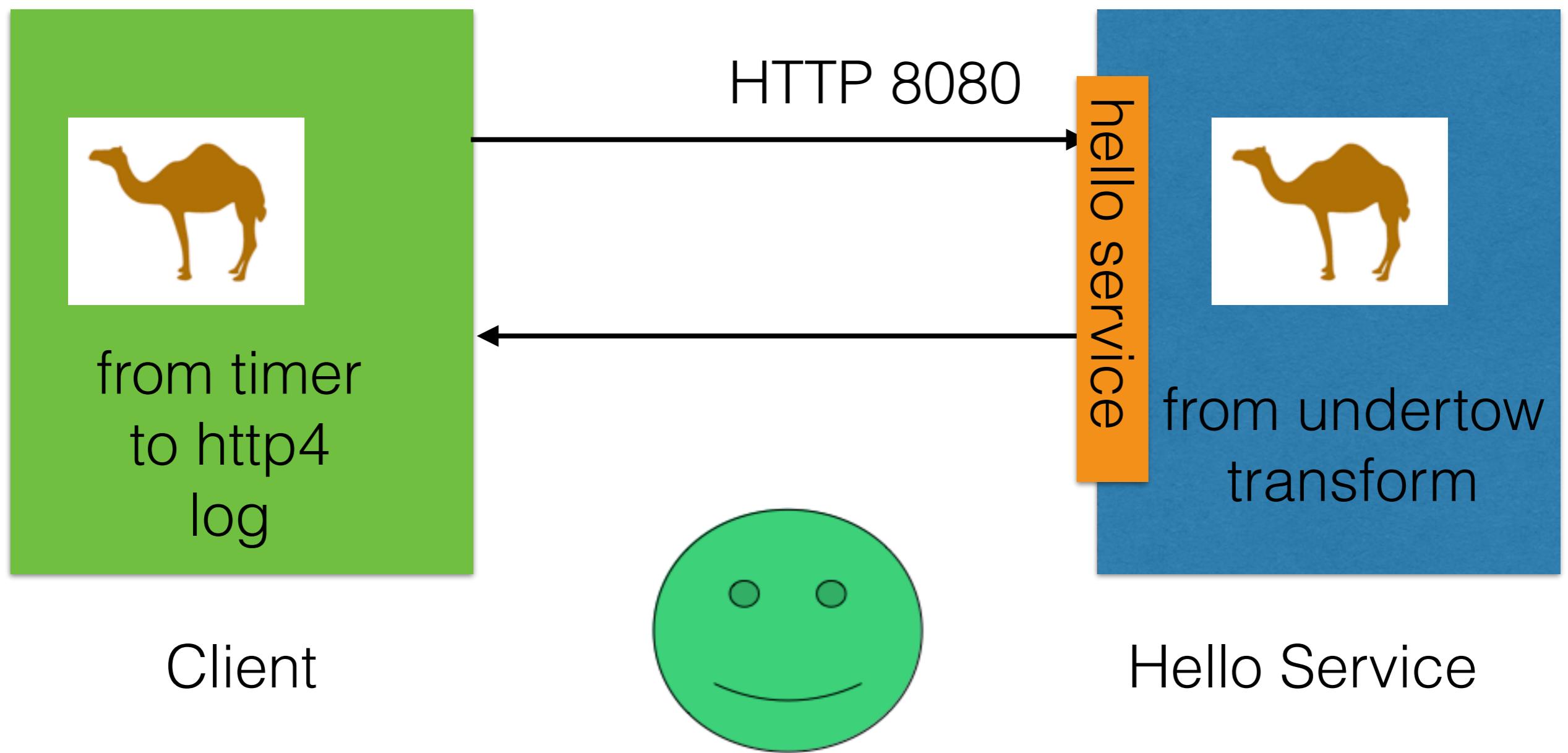
A code completion dropdown is open over the word "delay". The list includes:

- period long
- bridgeErrorHandler boolean
- daemon boolean
- delay** long
- c exceptionHandler org.apache.camel.spi.ExceptionHandler
- E exchangePattern org.apache.camel.ExchangePattern
- fixedRate boolean
- pattern java.lang.String
- repeatCount long
- synchronous boolean
- time java.util.Date
- c timer java.util.Timer

At the bottom of the dropdown, a note says: "Dot, space and some other keys will also close this lookup and be inserted into editor" followed by a small pi symbol.

<https://github.com/camel-idea-plugin/camel-idea-plugin>

Ready to run local

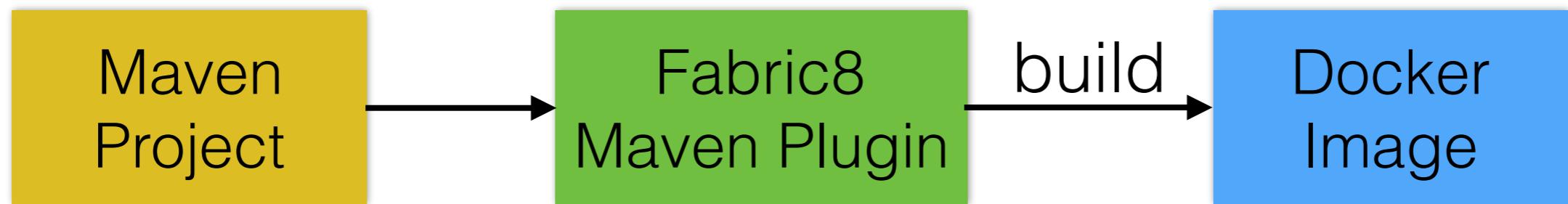


How to build Docker Image?

Maven
Project

Docker
Image

Fabric8 Maven Plugin



<https://maven.fabric8.io>

Fabric8 Maven Plugin



```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>fabric8-maven-plugin</artifactId>
  <version>3.3.5</version>
</plugin>
```

<https://maven.fabric8.io>

Installing Fabric8 Maven Plugin



```
mvn  
io.fabric8:fabric8-maven-plugin:3.3.5:setup
```

<https://maven.fabric8.io>

Fabric8 Maven Plugin



```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>fabric8-maven-plugin</artifactId>
  <version>3.3.5</version>
  <executions>
    <execution>
      <id>fmp</id>
      <goals>
        <goal>resource</goal>
        <goal>helm</goal>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Build Docker Image

OpenShift
S2I Build

```
[INFO] F8: Using OpenShift build with strategy S2I
[INFO] F8: Running generator wildfly-swarm
[INFO] F8: wildfly-swarm: Using Docker image fabric8/s2i-java:2.0 as base / builder
[INFO] Copying files to /Users/davsclaus/Documents/workspace/minishift-hello/helloswarm
[INFO] Building tar: /Users/davsclaus/Documents/workspace/minishift-hello/helloswarm/tar
[INFO] F8: [helloswarm:latest] "wildfly-swarm": Created docker source tar /Users/davsclau
rget/docker/helloswarm/latest/tmp/docker-build.tar
[INFO] F8: Creating BuildServiceConfig helloswarm-s2i for Source build
[INFO] F8: Creating ImageStream helloswarm
[INFO] F8: Starting Build helloswarm-s2i
[INFO] F8: Waiting for build helloswarm-s2i-1 to complete...
[INFO] F8: Receiving source from STDIN as archive ...
[INFO] F8: =====
[INFO] F8: Starting S2I Java Build .....
[INFO] F8: S2I binary build from fabric8-maven-plugin detected
[INFO] F8: Copying binaries from /tmp/src/maven to /deployments ...
[INFO] F8: ... done
```

mvn package fabric8:build

OpenShift S2I Build

```
[INFO] F8:  
[INFO] F8: Pushing image 172.30.1.1:5000/myproject/helloswarm:latest ...  
[INFO] F8: Pushed 0/23 layers, 0% complete  
[INFO] F8: Pushed 1/23 layers, 4% complete  
[INFO] F8: Pushed 2/23 layers, 9% complete  
[INFO] F8: Pushed 3/23 layers, 13% complete  
[INFO] F8: Pushed 4/23 layers, 17% complete  
[INFO] F8: Pushed 5/23 layers, 22% complete  
[INFO] F8: Pushed 6/23 layers, 26% complete  
[INFO] F8: Pushed 7/23 layers, 30% complete  
[INFO] F8: Pushed 8/23 layers, 35% complete  
[INFO] F8: Pushed 9/23 layers, 39% complete  
[INFO] F8: Pushed 10/23 layers, 43% complete  
[INFO] F8: Pushed 11/23 layers, 48% complete  
[INFO] F8: Pushed 12/23 layers, 52% complete  
[INFO] F8: Pushed 13/23 layers, 57% complete  
[INFO] F8: Pushed 14/23 layers, 61% complete  
[INFO] F8: Pushed 15/23 layers, 65% complete  
[INFO] F8: Pushed 16/23 layers, 70% complete  
[INFO] F8: Pushed 17/23 layers, 74% complete  
[INFO] F8: Pushed 18/23 layers, 78% complete  
[INFO] F8: Pushed 19/23 layers, 83% complete  
[INFO] F8: Pushed 20/23 layers, 87% complete  
[INFO] F8: Pushed 21/23 layers, 91% complete  
[INFO] F8: Pushed 22/23 layers, 96% complete  
[INFO] F8: Pushed 23/23 layers, 100% complete  
[INFO] F8: Build helloswarm-s2i-1 Complete  
[INFO] F8: Found tag on ImageStream helloswarm tag: sha256:f5fb6be5e26113967b6bd:  
[INFO] F8: ImageStream helloswarm written to /Users/davsclaus/Documents/workspace  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----
```

Live
Build Log
Output

OpenShift S2I Build

The screenshot shows the OpenShift web interface for a project named "My Project". The left sidebar contains navigation links for Home, Overview, Applications, Builds, Resources, and Storage. The main content area displays a build named "helloswarm-s2i" created 3 minutes ago. The build has labels: group, com.foo, project, helloswarm, provider, fabric8, and More labels... The History tab is selected, showing a single completed build (#1) started 3 minutes ago. The build status is "Complete" and it took 1 minute, 14 seconds. There is also a configuration and environment section.

Project
My Project

Add to project

Builds » helloswarm-s2i

helloswarm-s2i created 3 minutes ago

group com.foo project helloswarm provider fabric8 More labels...

Overview Applications Builds Resources Storage

History Configuration Environment

✓ Build #1 is complete. [View Log](#)
started 3 minutes ago

Filter by label Add

Build	Status	Duration
#1	✓ Complete	1 minute, 14 seconds

MiniShift

Docker Repository

TIP Run minishift docker-env
to setup Docker CLI

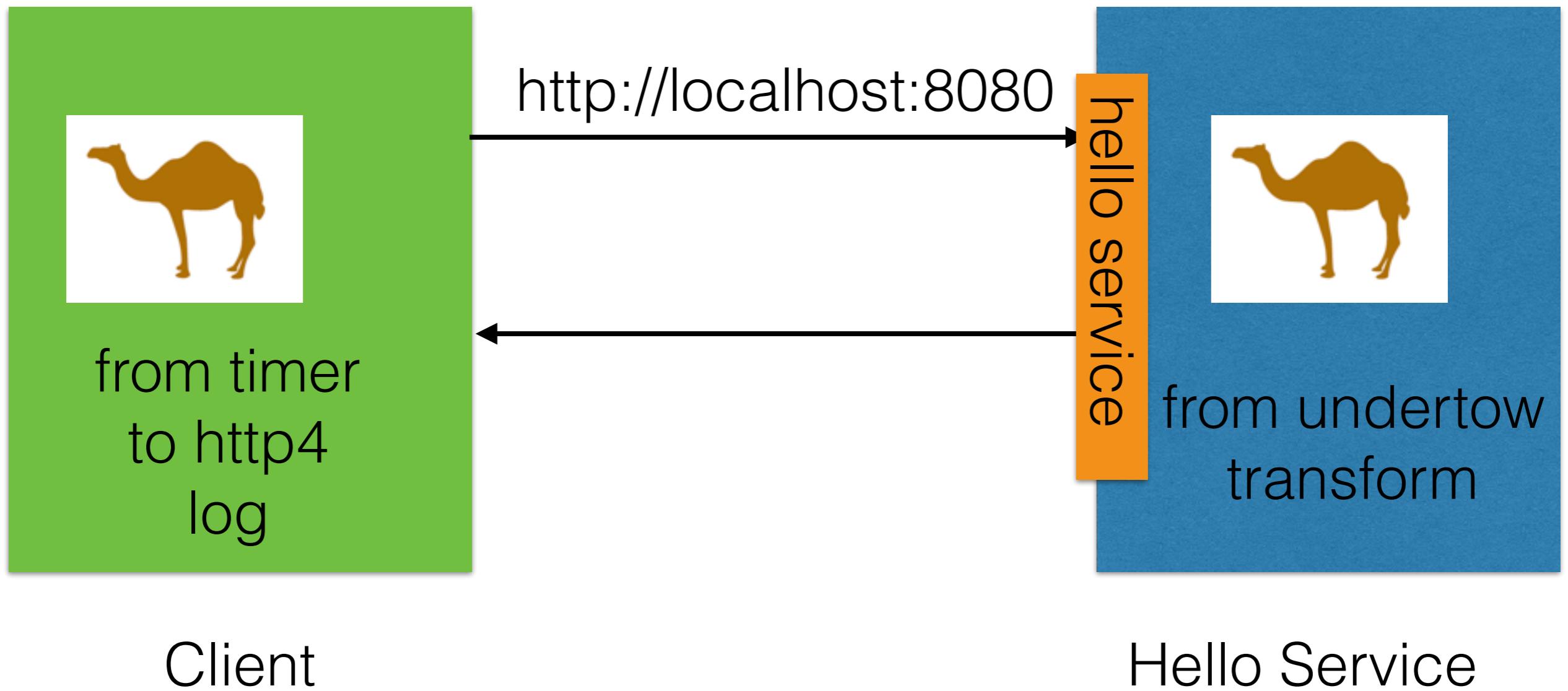
```
$ docker images
```

REPOSITORY	Client	TAG
172.30.1.1:5000/myproject/client		latest
172.30.1.1:5000/myproject/helloswarm		latest
openshift/origin-sti-builder		v1.5.0
openshift/origin-deployer		v1.5.0
openshift/origin-docker-registry		v1.5.0
openshift/origin-haproxy-router		v1.5.0
openshift/origin		v1.5.0
openshift/origin-pod		v1.5.0
fabric8/s2i-java		2.0

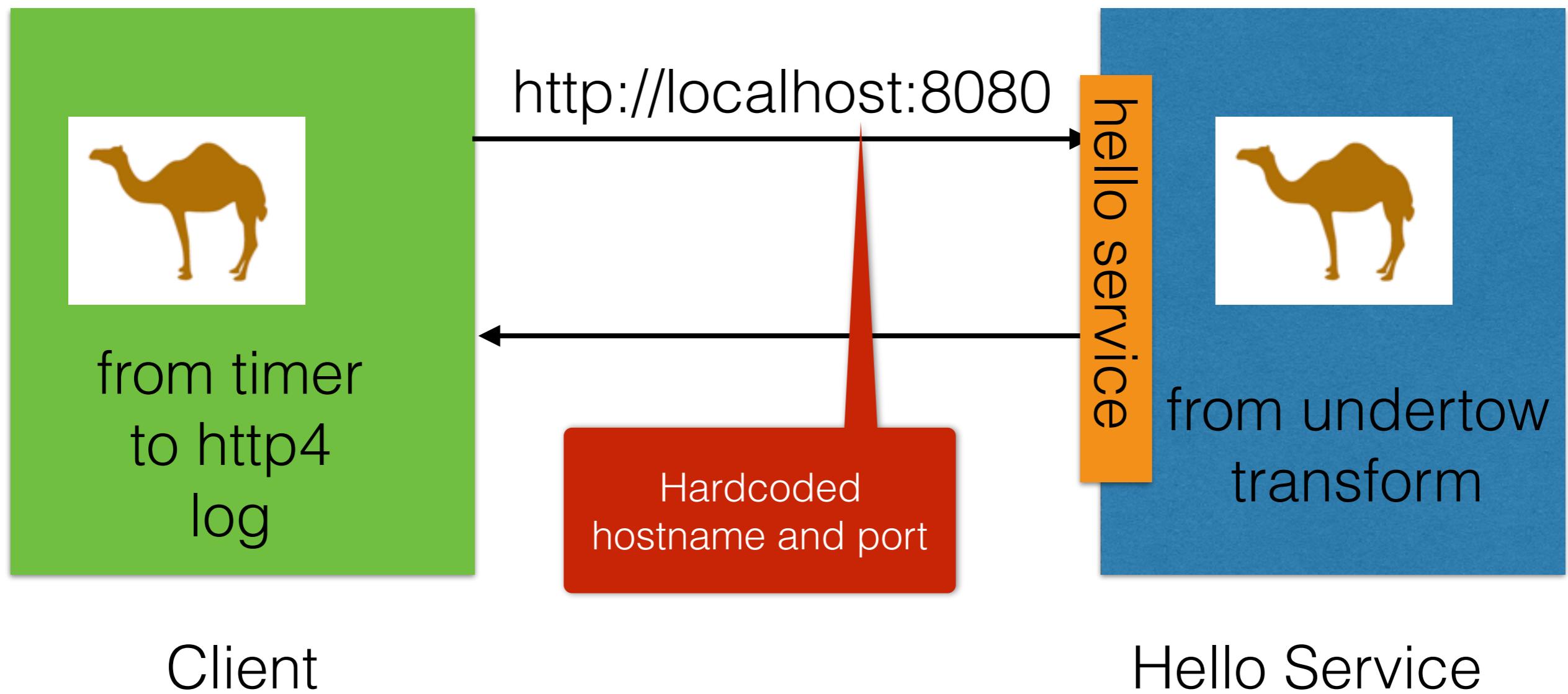
Client

Hello Service

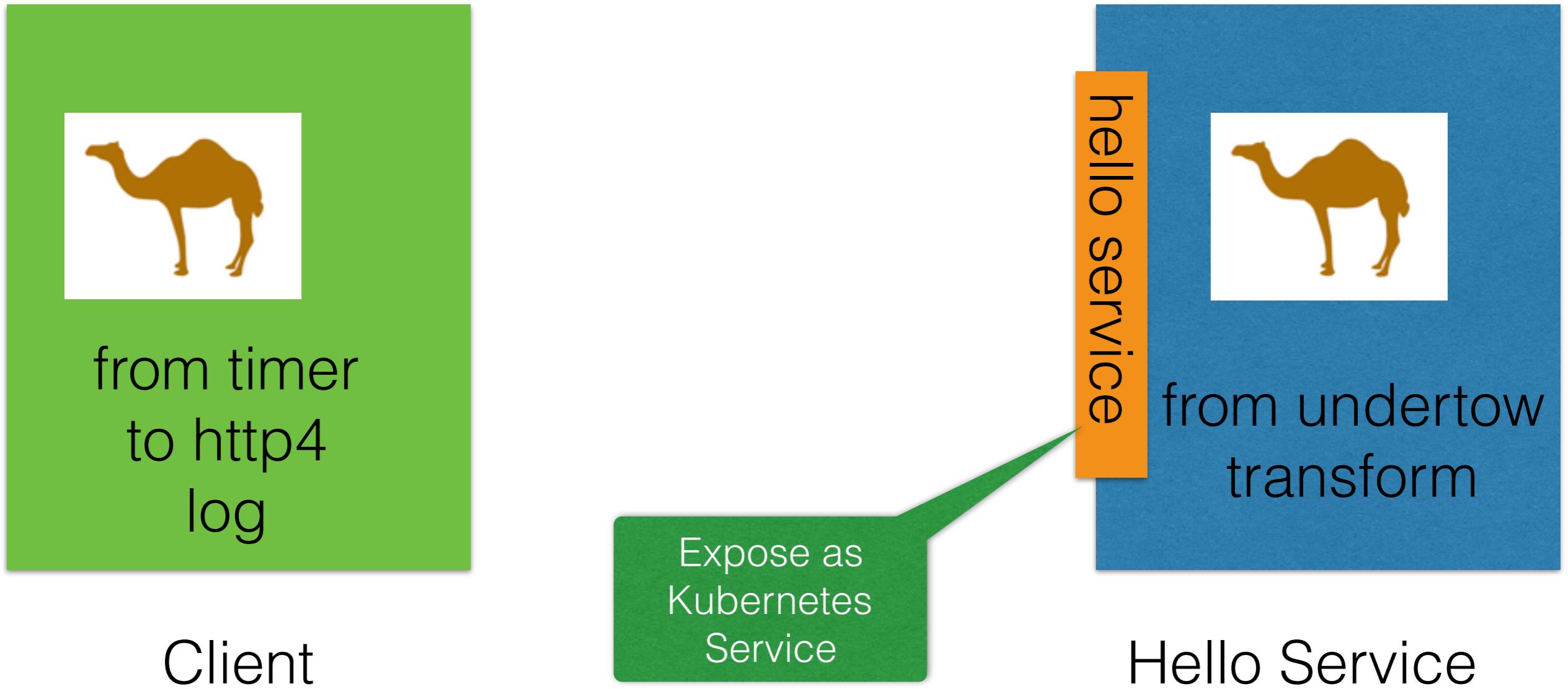
Our Demo



Static vs Dynamic Platform



Dynamic Platform



Dynamic Platform

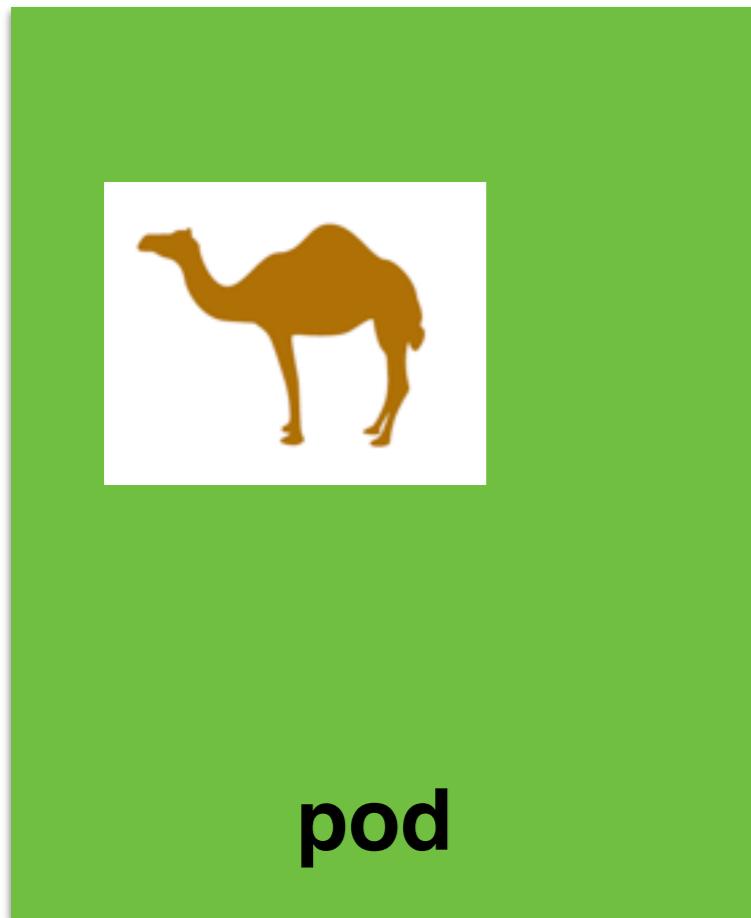


Client

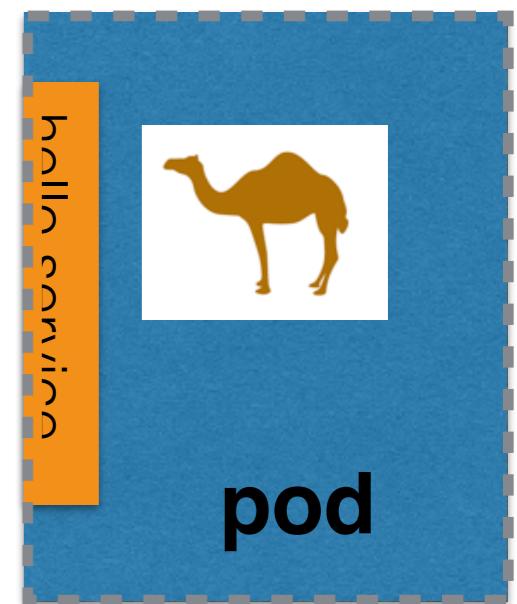


Hello Service

Dynamic Platform



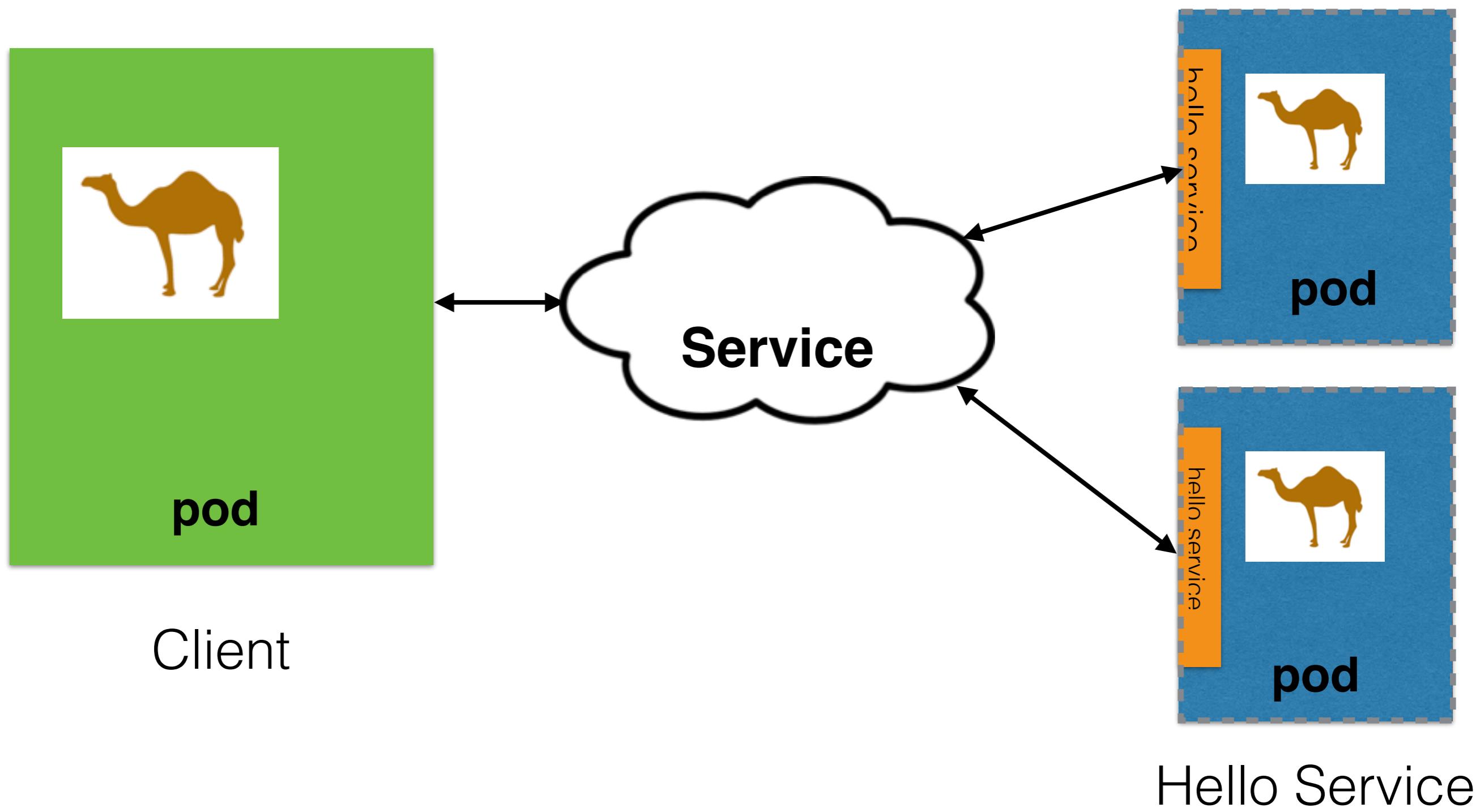
Client



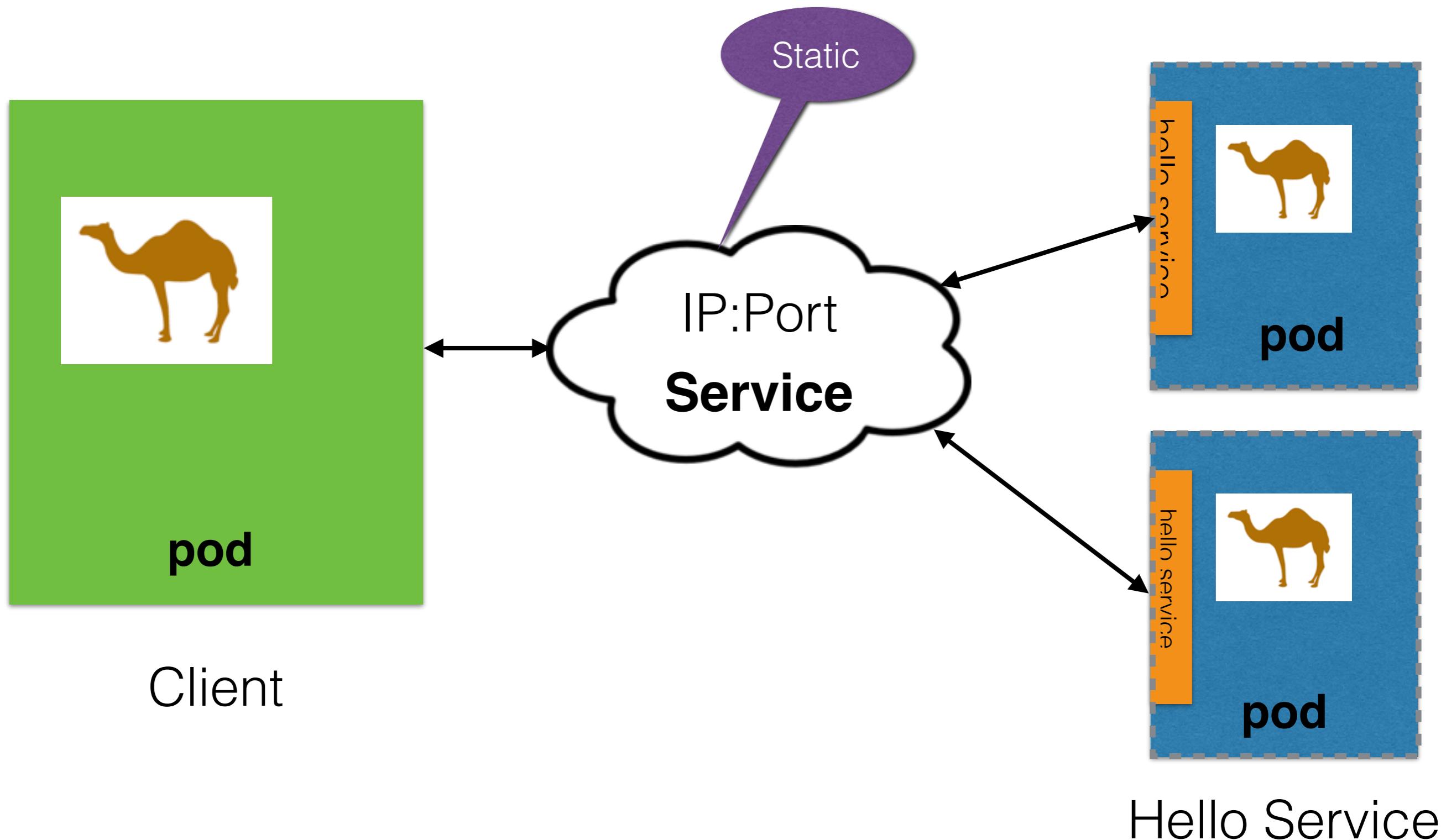
Hello Service



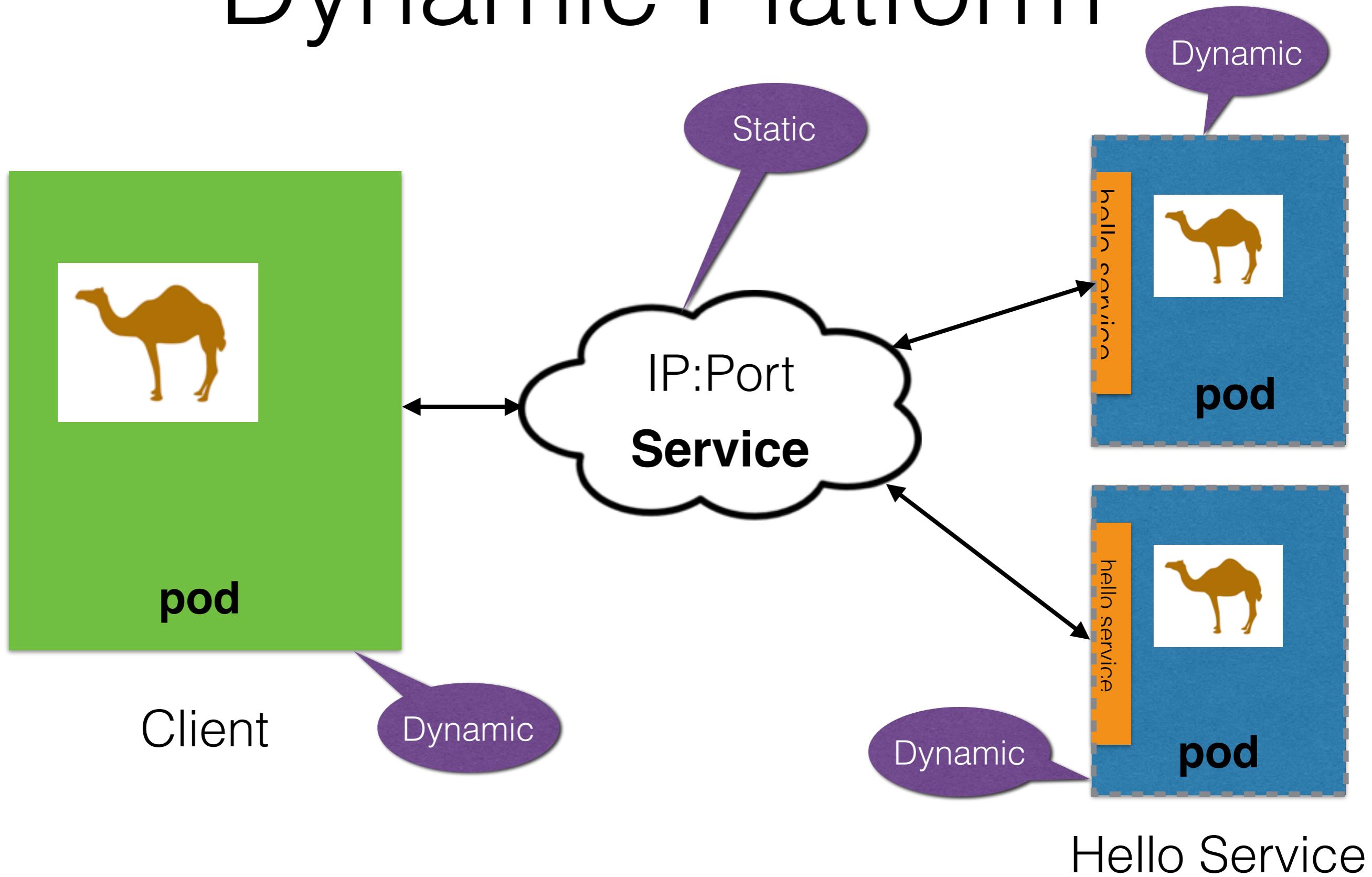
Dynamic Platform



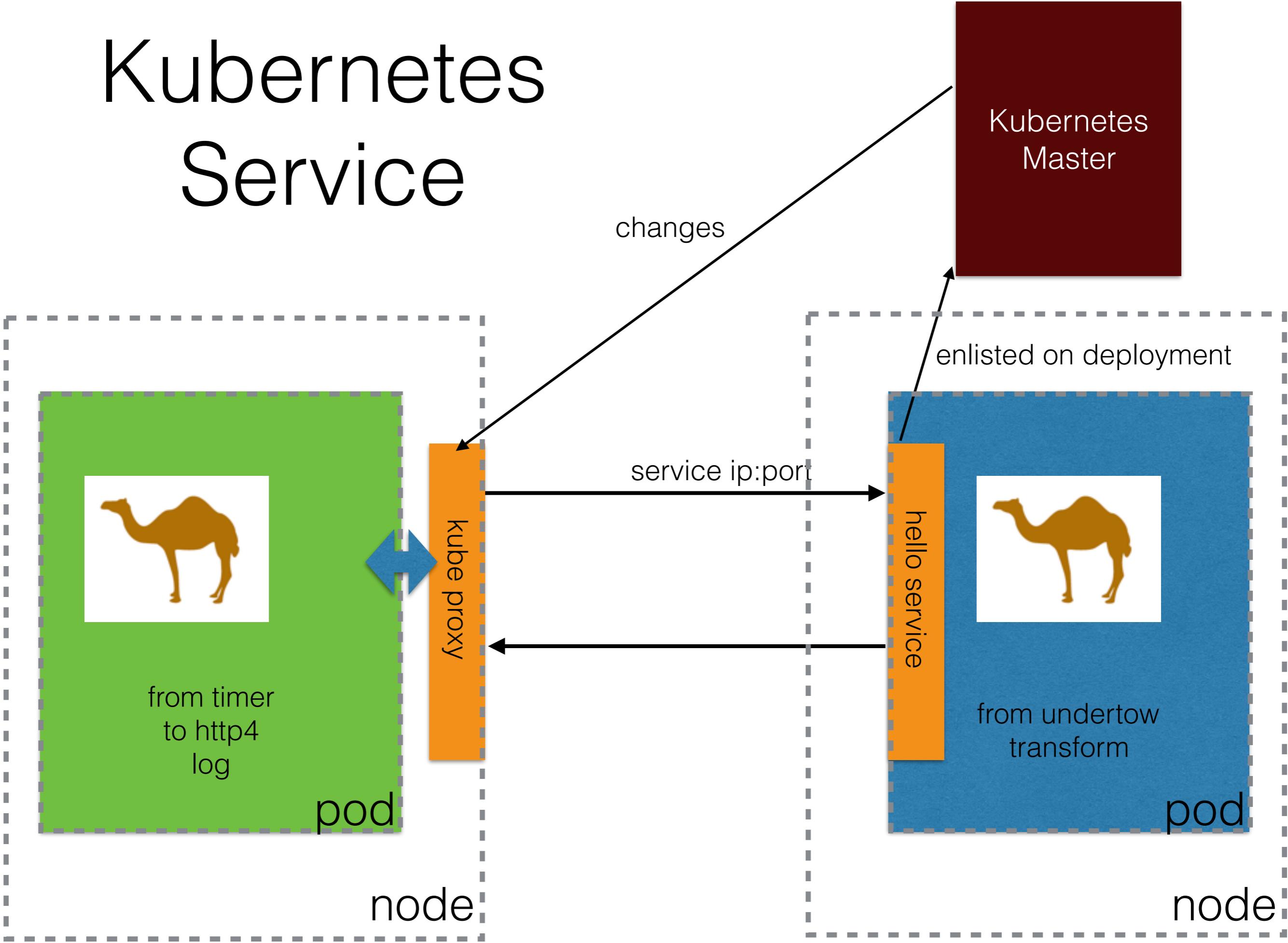
Dynamic Platform



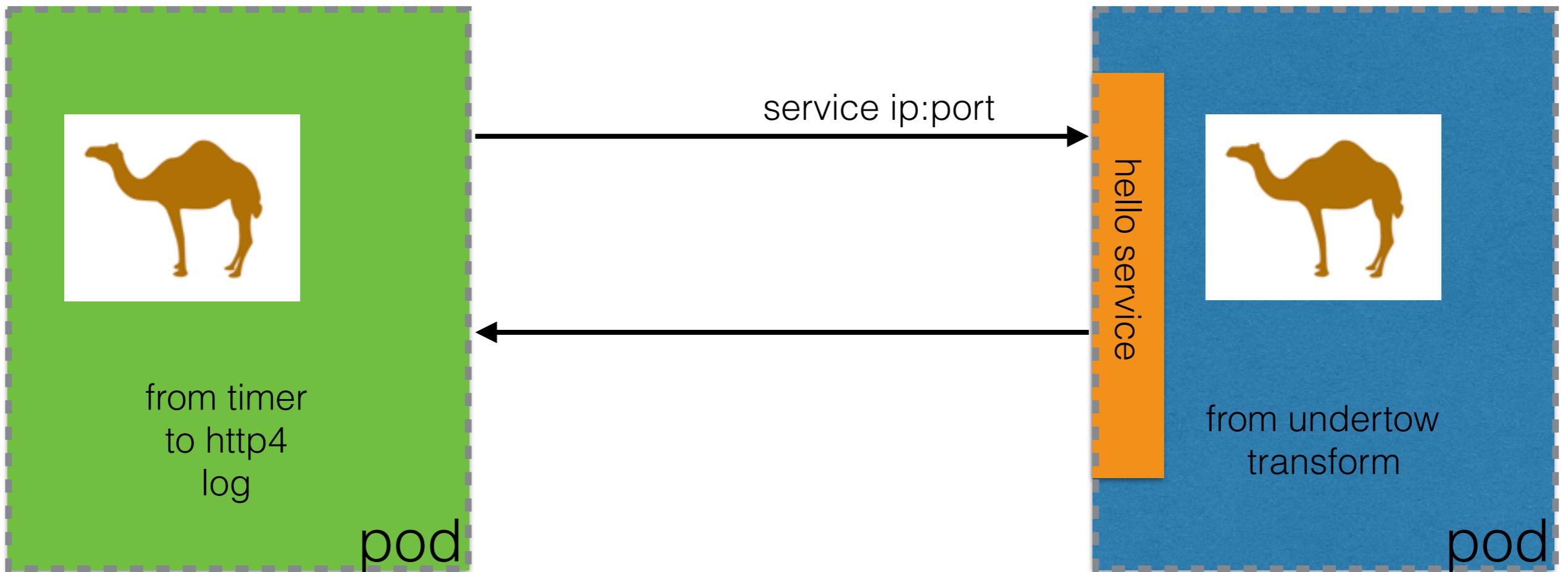
Dynamic Platform



Kubernetes Service



Kubernetes Service from user point of view



Using Kubernetes Service



from timer
to http4
log

Client

We want to use hello service

How do we do that?

Using Kubernetes Service



from timer
to http4
log

- Environment Variables

Client

- Hostname
- Port

```
export HELLOSWARM_SERVICE_HOST="172.30.237.105"  
export HELLOSWARM_SERVICE_PORT="8080"
```

Service Discovery using DNS is also available

Service using ENV



from timer
to http4
log

- {{service:name}}

Client

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from(uri: "timer:too?period=2000")
            .to("http4: {{service:helloswarm}}/hello")
            .log("${body}");
    }
}
```

Service using DNS



from timer
to http4
log

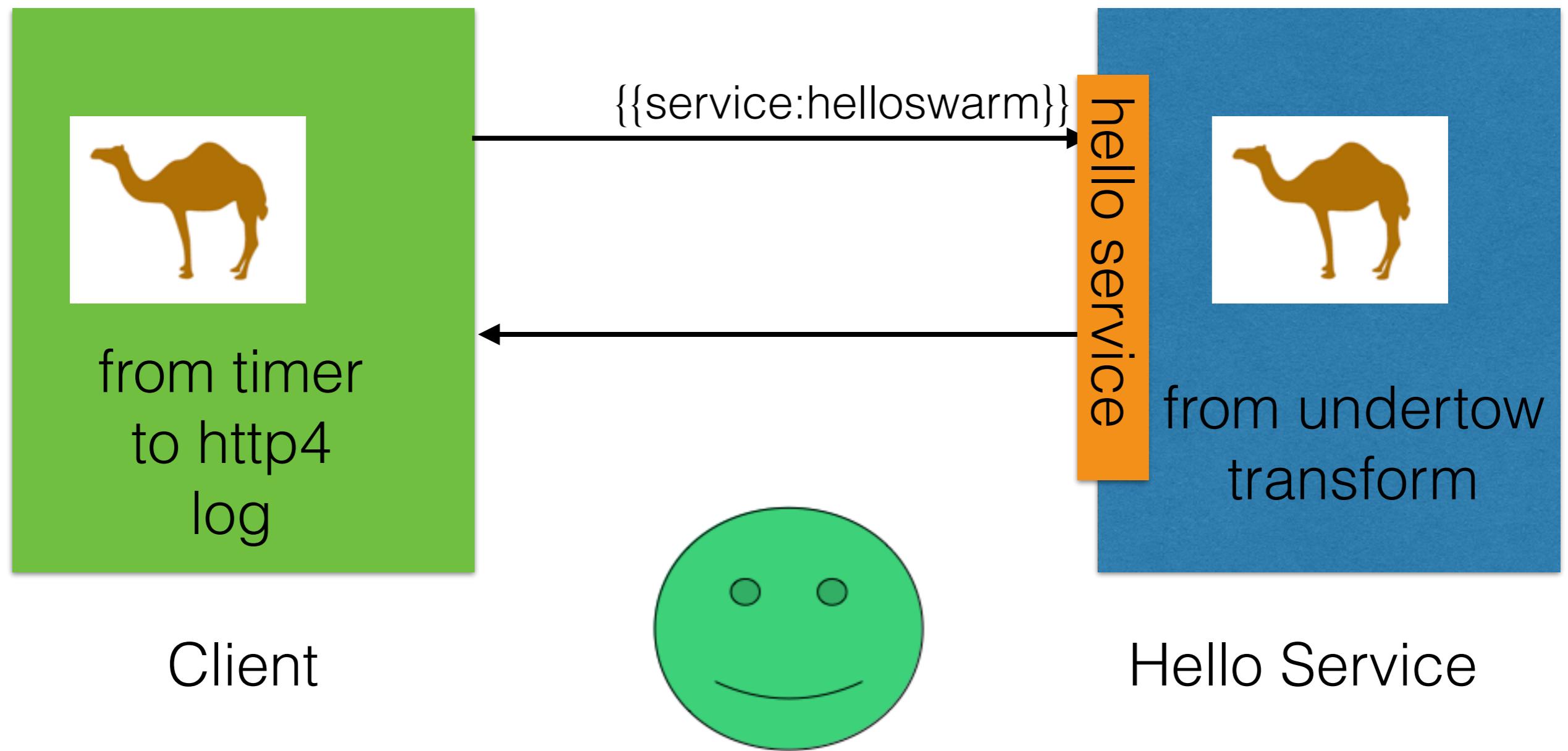
- servicename:port

Client

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer:too?period=2000")
            .to("http4helloworld:8080/hello")
            .log("${body}");
    }
}
```

Ready to run in OpenShift



How to deploy to OpenShift?

Maven Project



How to deploy to OpenShift?



Deploy - Hello Service

hello service



from undertow
transform

- mvn fabric8:deploy

Hello Service

```
davsclaus:/Users/davsclaus/Documents/workspace/minishift-hello/helloswarm (master)$ mvn fabric8:deploy
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Wildfly Swarm Example 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] >>> fabric8-maven-plugin:3.3.5:deploy (default-cli) > install @ helloswarm >>>
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ helloswarm ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- fabric8-maven-plugin:3.3.5:resource (fmp) @ helloswarm ---
[INFO] F8: Running in OpenShift mode
[INFO] F8: Using docker image name of namespace: myproject
[INFO] F8: Running generator wildfly-swarm
[INFO] F8: wildfly-swarm: Using Docker image fabric8/java-jboss-openjdk8-jdk:1.2 as base / builder
[INFO] F8: fmp-controller: Adding a default Deployment
[WARNING] F8: fmp-service: Implicit service port mapping to port 80 has been disabled for the used port
either use set the config port = 80 or use legacyPortMapping = true. See https://maven.fabric8.io/#fmp-s
[INFO] F8: fmp-service: Adding a default service 'helloswarm' with ports [8080]
```

Deploy - Client



from timer
to http4
log

- mvn fabric8:deploy

Client

```
davsclaus:/Users/davsclaus/Documents/workspace/minishift-hello/client (master) $ mvn fabric8:deploy
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building client 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] >>> fabric8-maven-plugin:3.3.5:deploy (default-cli) > install @ client >>>
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ client ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO]
[INFO] --- fabric8-maven-plugin:3.3.5:resource (fmp) @ client ---
[INFO] F8: Running in OpenShift mode
[INFO] F8: Using docker image name of namespace: myproject
[INFO] F8: Running generator spring-boot
[INFO] F8: spring-boot: Using Docker image fabric8/java-jboss-openjdk8-jdk:1.2 as base / builder
[INFO] F8: fmp-controller: Adding a default Deployment
[WARNING] F8: fmp-service: Implicit service port mapping to port 80 has been disabled for the used port
either use set the config port = 80 or use legacyPortMapping = true. See https://maven.fabric8.io/#fmp-
[INFO] F8: fmp-service: Adding a default service 'client' with ports [8080]
[INFO] F8: spring-boot-health-check: Adding readiness probe on port 8080 path='/health' scheme='HTTP'
```

Run - Client



Runs in foreground, tail log,
undeploys when ctrl + c

from timer
to http4
log

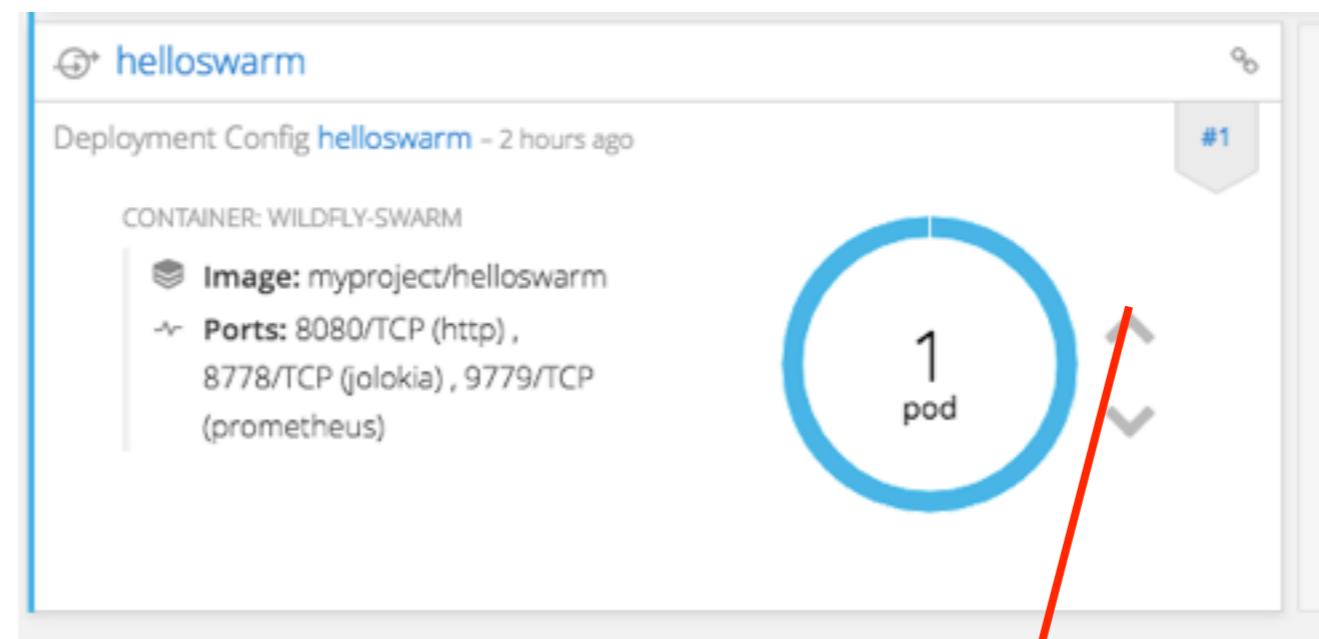
- mvn fabric8:run

Client

```
[INFO] Updated DeploymentConfig: target/fabric8/applyJson/myproject/deploymentconfig-client-3.json
[INFO] F8: HINT: Use the command `oc get pods -w` to watch your pods start up
[INFO] F8: Scaling DeploymentConfig myproject/client to replicas: 1
[INFO] F8: Watching pods with selector LabelSelector(matchExpressions=[], matchLabels={project=client, 
erties={}}) waiting for a running pod...
[INFO] F8:[NEW] client-2-4vrdk status: Running Ready
[INFO] F8:[NEW] Tailing log of pod: client-2-4vrdk
[INFO] F8:[NEW] Press Ctrl-C to scale down the app and stop tailing the log
[INFO] F8:[NEW]
[INFO] F8: Starting the Java application using /opt/run-java/run-java.sh ...
[INFO] F8: exec java -javaagent:/opt/jolokia/jolokia.jar=config=/opt/jolokia/etc/jolokia.properties -c
[INFO] F8: I> No access restrictor found, access to any MBean is allowed
[INFO] F8: Jolokia: Agent started with URL https://172.17.0.6:8778/jolokia/
[INFO] F8:
[INFO] F8:
[INFO] F8: . __
[INFO] F8: \ / ' - - - ( ) _ _ _ - \ \ \
[INFO] F8: ( ( ) \ _ | ' _ | ' _ | ' _ \ _ | \ \ \
[INFO] F8: \ \ _ ) | | | | | | | ( | | ) ) )
[INFO] F8: ' | _ | . | | | | | | | , | / / /
[INFO] F8: ===== | | ===== | | / = / / /
[INFO] F8: :: Spring Boot ::      (v1.5.3.RELEASE)
[INFO] F8:
```

Scaling

- Change deployment replicas



Load balancing
is random

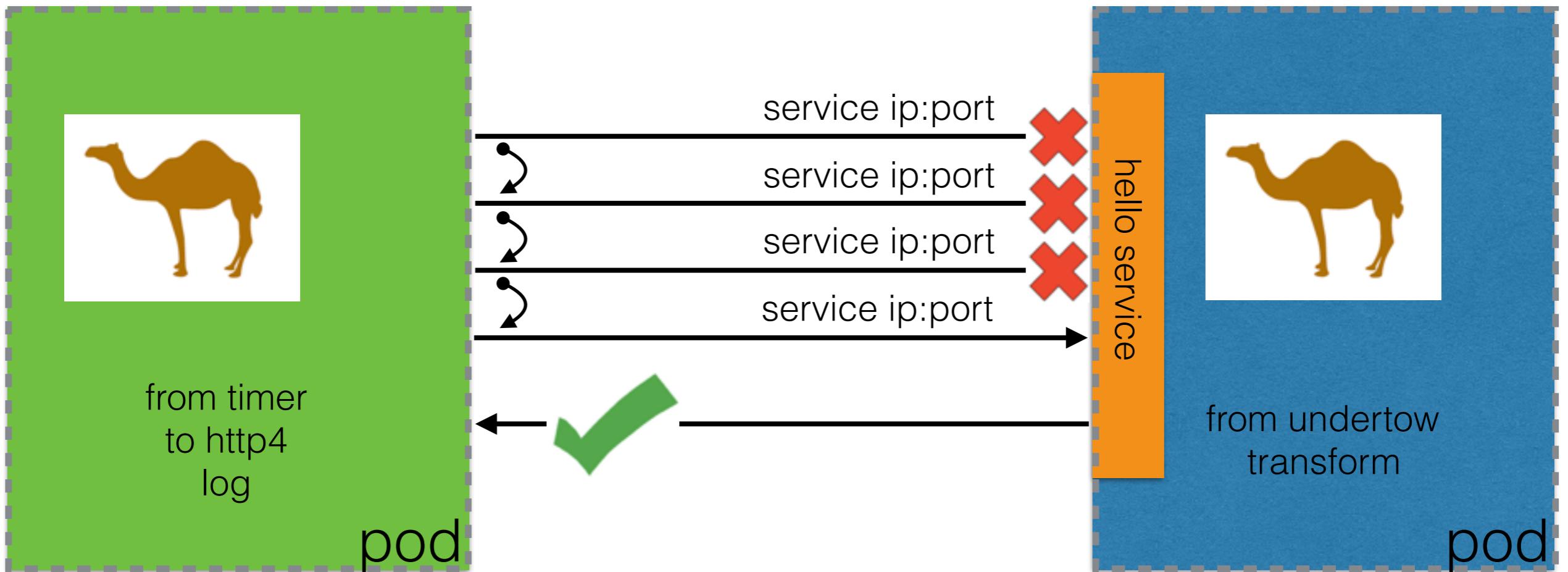
Scaling

- Service load balancing

```
: Swarm says hello from helloswarm-1-bjdbj
: Swarm says hello from helloswarm-1-s5rfh
: Swarm says hello from helloswarm-1-s5rfh
: Swarm says hello from helloswarm-1-bjdbj
: Swarm says hello from helloswarm-1-s5rfh
: Swarm says hello from helloswarm-1-s5rfh
: Swarm says hello from helloswarm-1-bjdbj
: Swarm says hello from helloswarm-1-s5rfh
: Swarm says hello from helloswarm-1-bjdbj
```

Error Handling

- Client Side Retry



Error Handling

- Client Side Retry

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        // try to call the service again
        onException(Exception.class)
            .maximumRedeliveries(10)
            .redeliveryDelay(1000);

        from( uri: "timer:foo?period=2000")
            .to("http4:{{service:helloswarm}}/hello")
            .log("${body}");
    }
}
```

Error Handling

- Client Side Retry

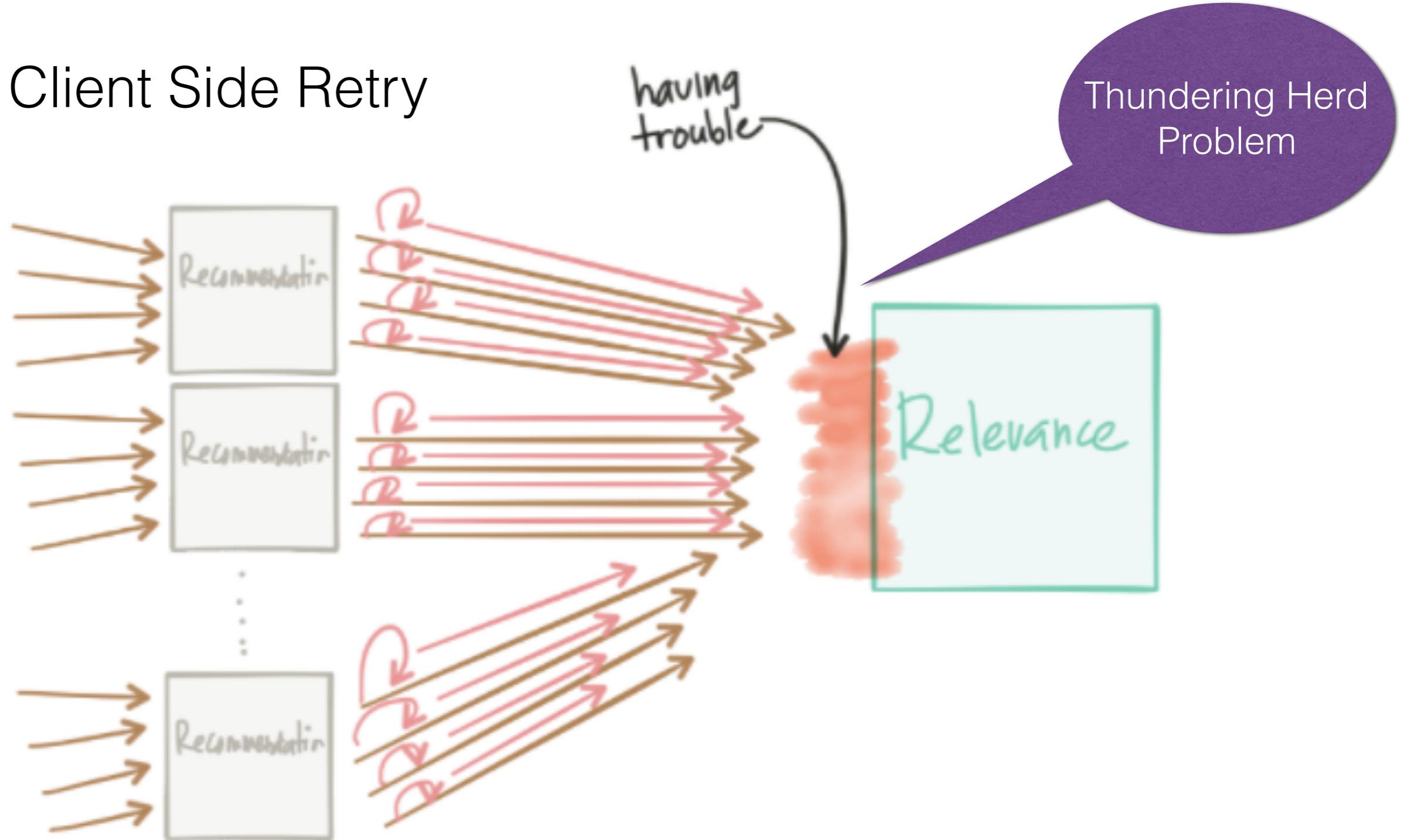
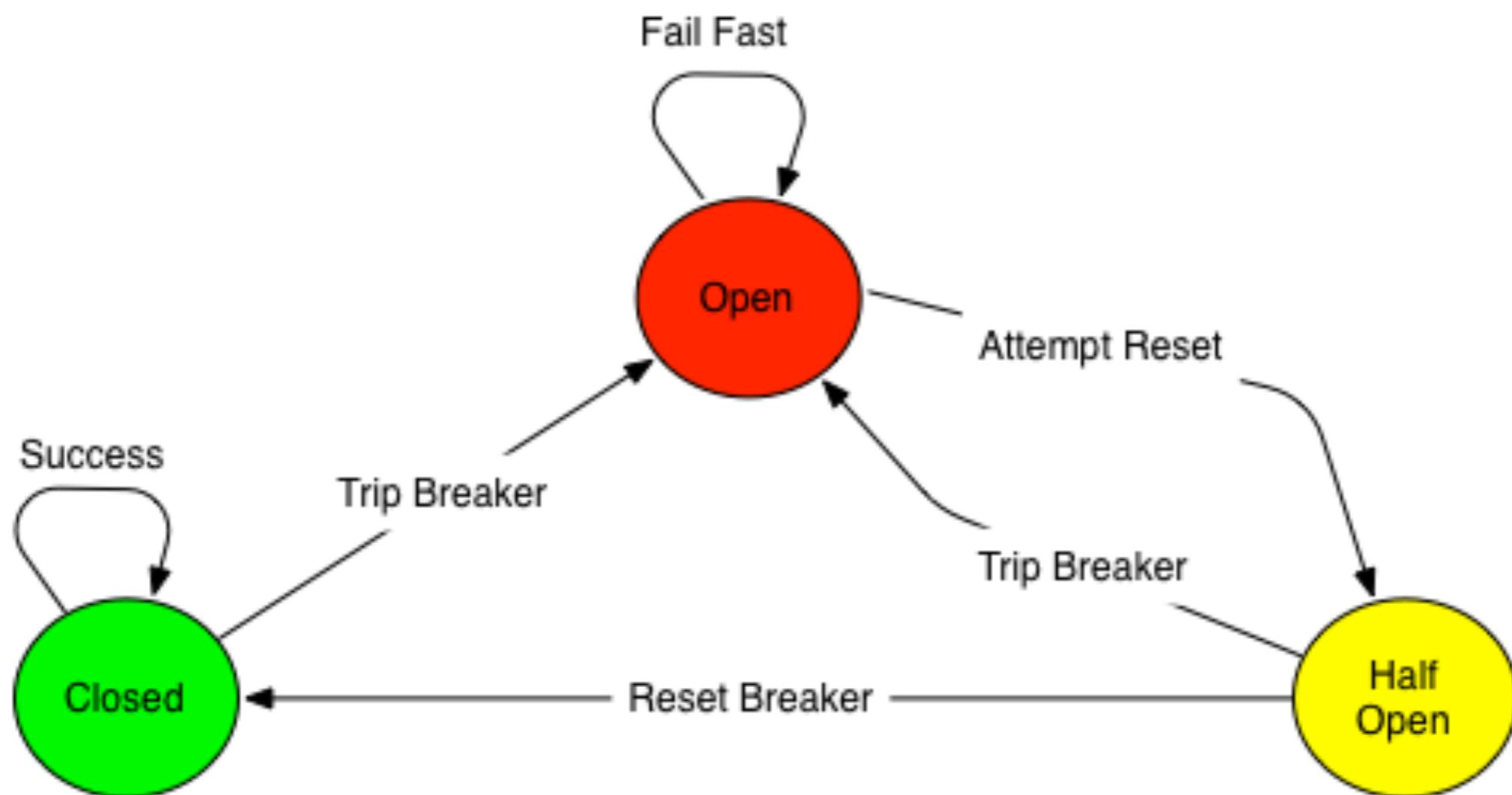


Figure by Christian Posta

Error Handling

- Client Side Circuit Breaker with Hystrix



Error Handling

- Client Side Circuit Breaker with Hystrix

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from( uri: "timer:foo?period=2000")
            .hystrix()
            .to("http4:{{service:helloswarm}}/hello")
            .onFallback()
            .setBody().constant( value: "Nobody want to talk to me")
            .end()
            .log("${body}");
    }
}
```

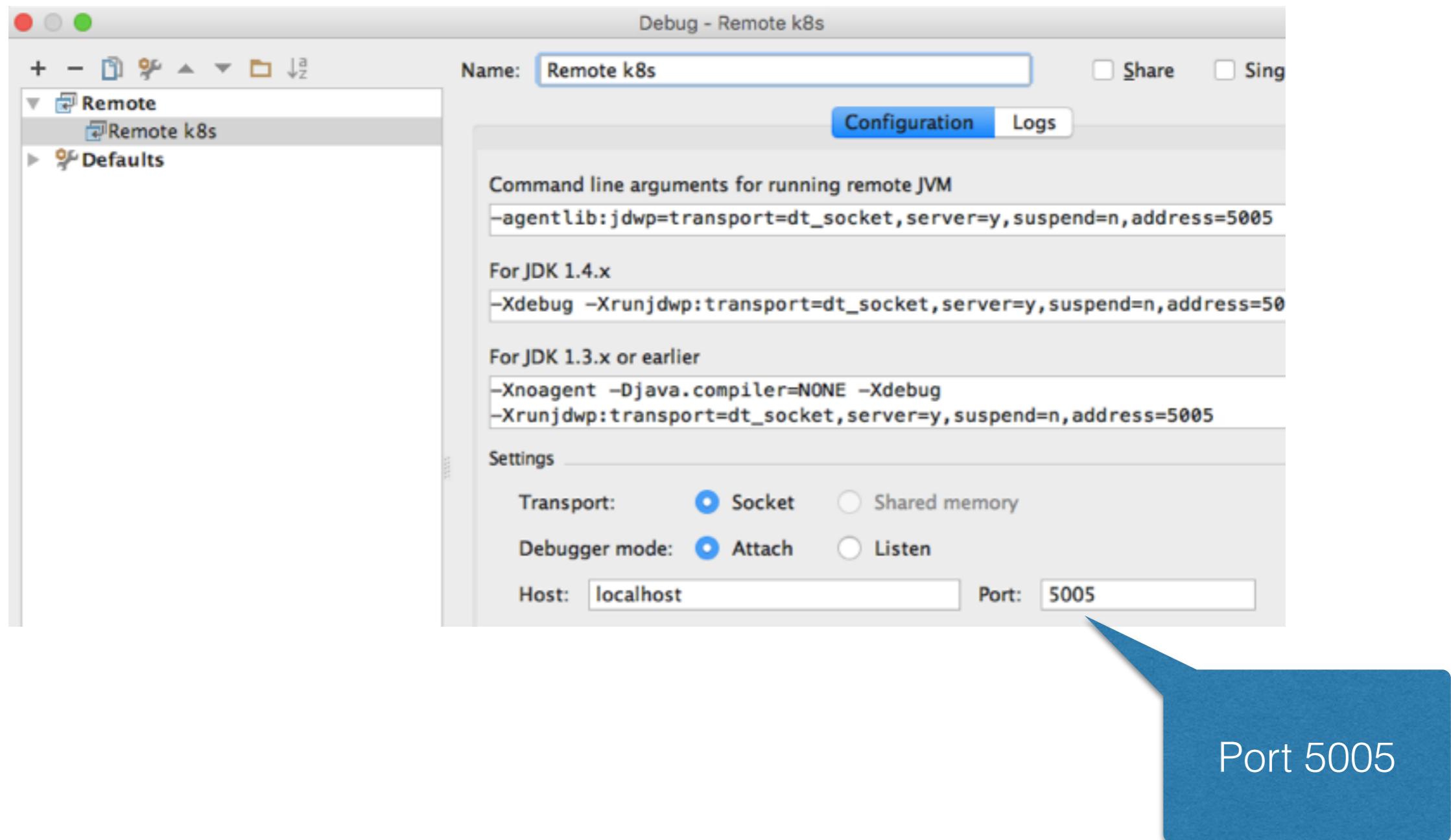
Debugging

- Debugging Pods

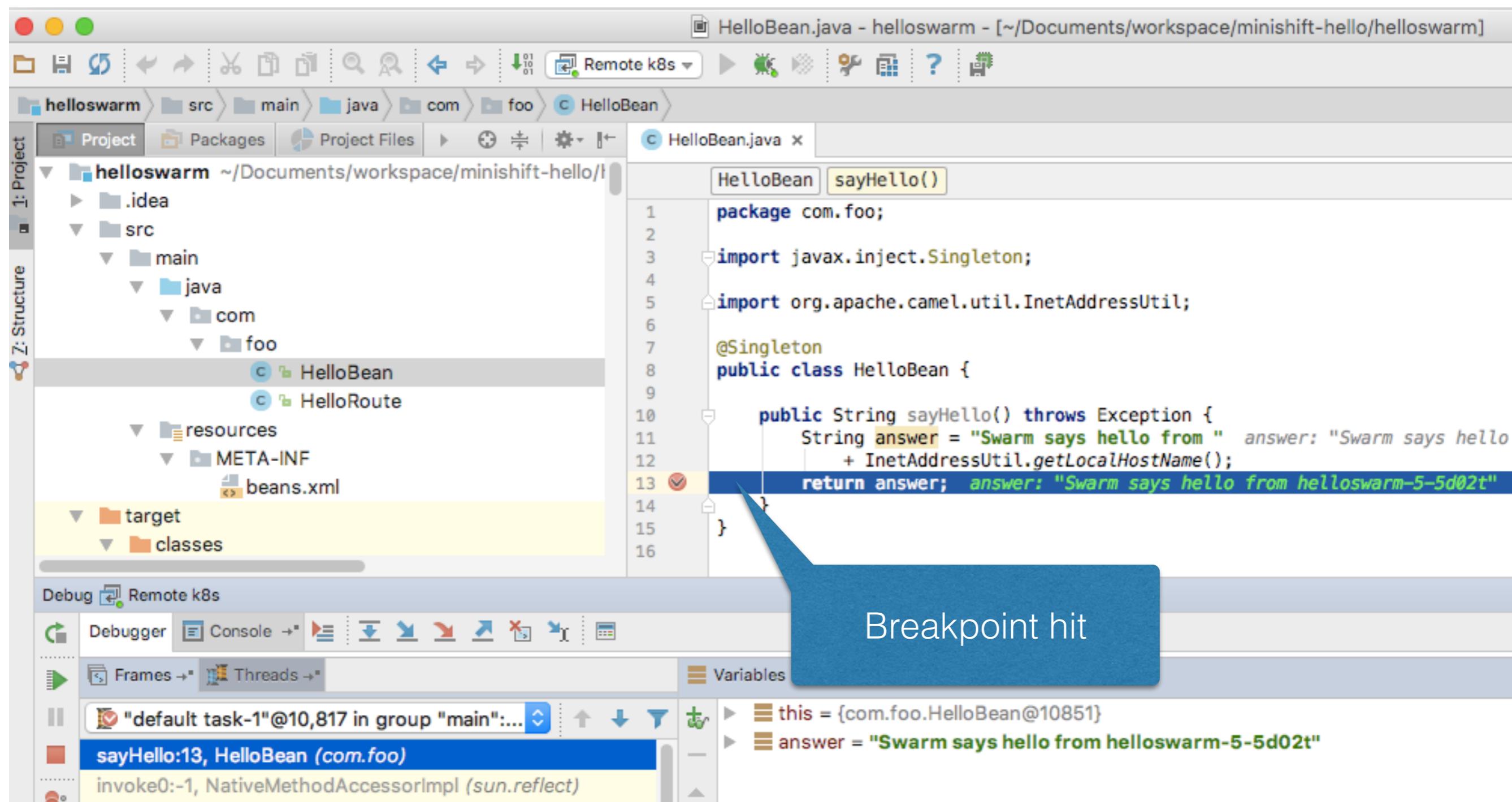
```
mvn fabric8:debug
```

```
[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



OpenShift CLI

You can also use CLI from
docker &
kubernetes

- oc get pods

```
davsclaus:/Users/davsclaus/$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
client-2-j7142	1/1	Running	0	4m
client-s2i-1-build	0/1	Completed	0	10m
client-s2i-2-build	0/1	Completed	0	4m
helloswarm-1-bjdbj	1/1	Running	2	16m
helloswarm-1-s5rfh	1/1	Running	0	22m
helloswarm-s2i-1-build	0/1	Completed	0	28m
helloswarm-s2i-2-build	0/1	Completed	0	24m

OpenShift CLI

- `oc logs -f <pod name>`

```
davsclaus:/Users/davsclaus/Documents/workspace/minishift-hello (master)$ oc logs -f client-2-7fnn9
Starting the Java application using /opt/run-java/run-java.sh ...
exec java -javaagent:/opt/jolokia/jolokia.jar=config=/opt/jolokia/etc/jolokia.properties -cp . -jar
I> No access restrictor found, access to any MBean is allowed
Jolokia: Agent started with URL https://172.17.0.5:8778/jolokia/
```

The Spring Boot logo is a stylized graphic composed of various characters including '^', 'v', '(', ')', '[', ']', '{', '}', '=', '+', and commas. It features a central vertical column of '^' and 'v' characters, flanked by parentheses and brackets. The base of the logo is a horizontal line with a central '=' sign, and a diagonal line extends from the right side of the base towards the right edge of the logo area.

OpenShift CLI

- oc get service

```
davsclaus:/Users/davsclaus/$ oc get services
```

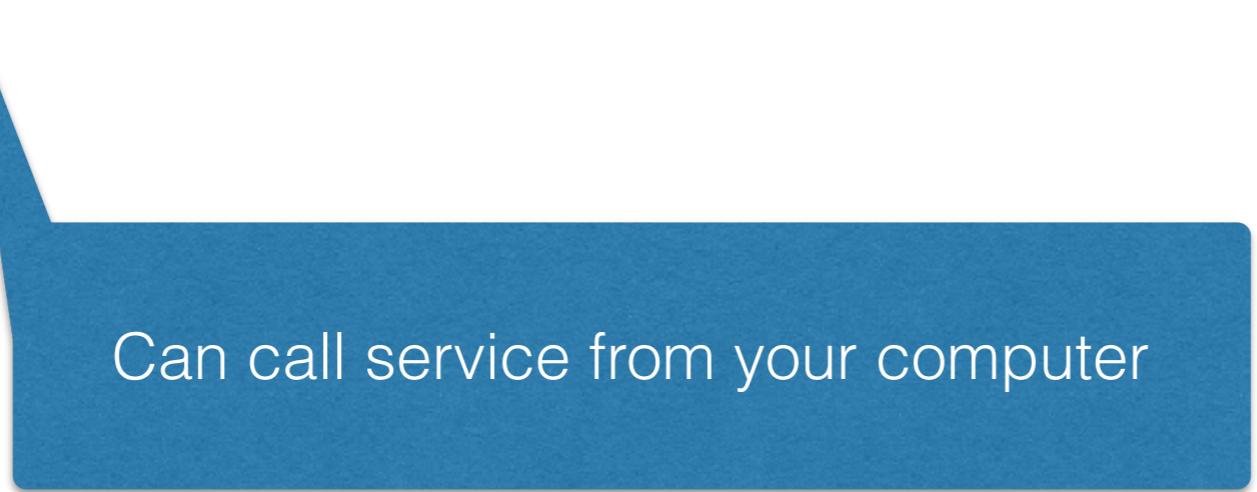
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
client	172.30.232.162	<none>	8080/TCP	11m
helloswarm	172.30.167.101	<none>	8080/TCP	23m

OpenShift CLI

- oc get routes

```
davsclaus:/Users/davsclaus/$ oc get routes
```

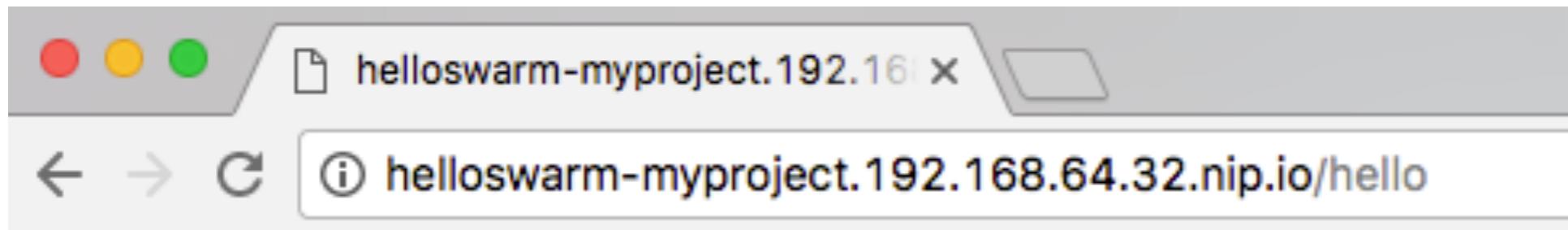
NAME	HOST/PORT	PATH	SERVICES	PORT
client	client-myproject.192.168.64.32.nip.io		client	8080
helloswarm	helloswarm-myproject.192.168.64.32.nip.io		helloswarm	8080



Can call service from your computer

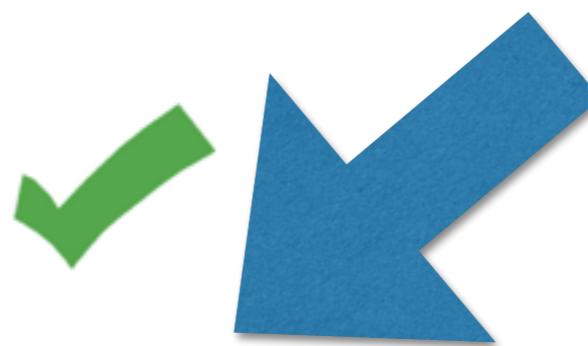
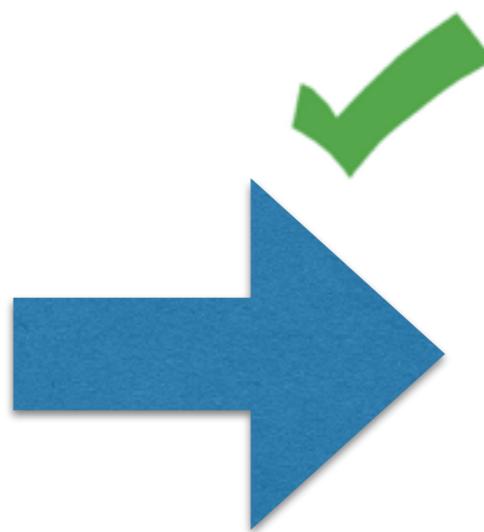
Access Service from your computer

- minishift openshift service hellowarm -n myproject

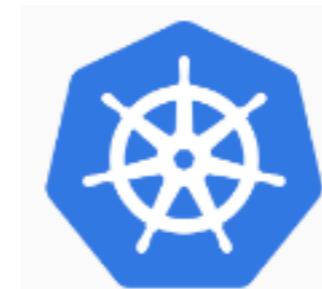


Name of project
(namespace)

Key Message



or



Tip of Iceberg

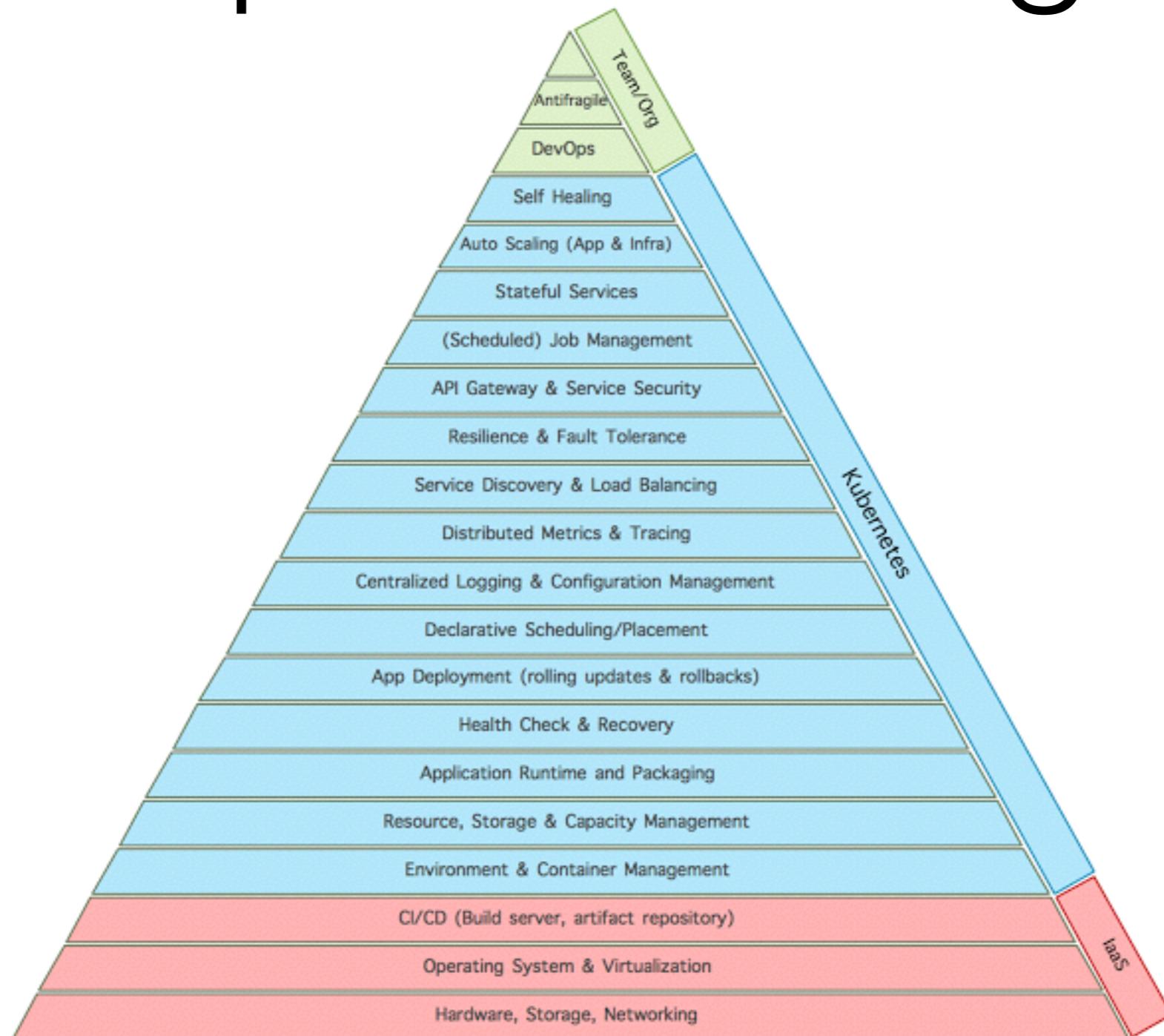
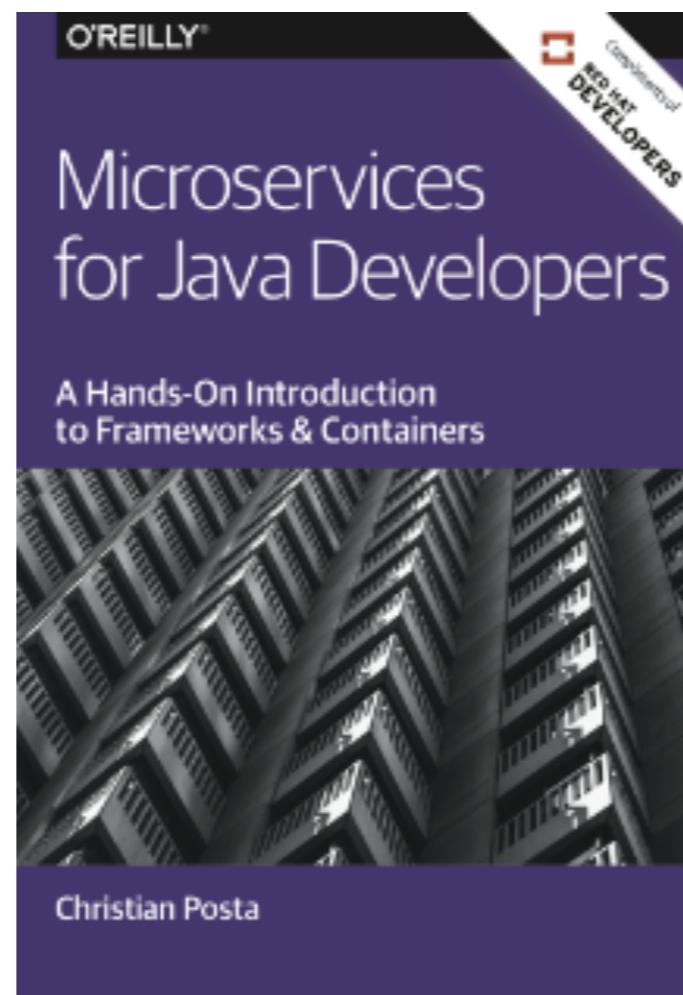


Figure by Bilgin Ibryam

More Details



<http://developers.redhat.com/promotions/microservices-for-java-developers/>

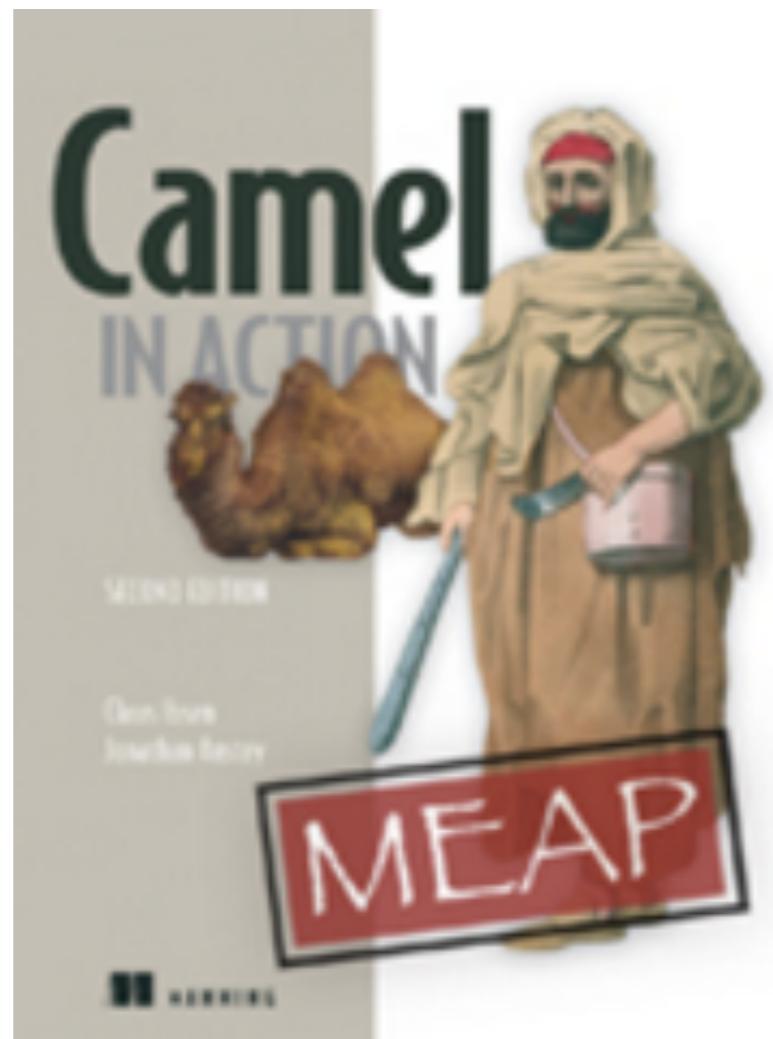
More Details



Grant Shipley & Graham Dumpleton

<https://www.openshift.com/promotions/for-developers.html>

Shameless Promotion



Coupon code:
camel39
gives 39% discount

<http://manning.com/ibsen2>



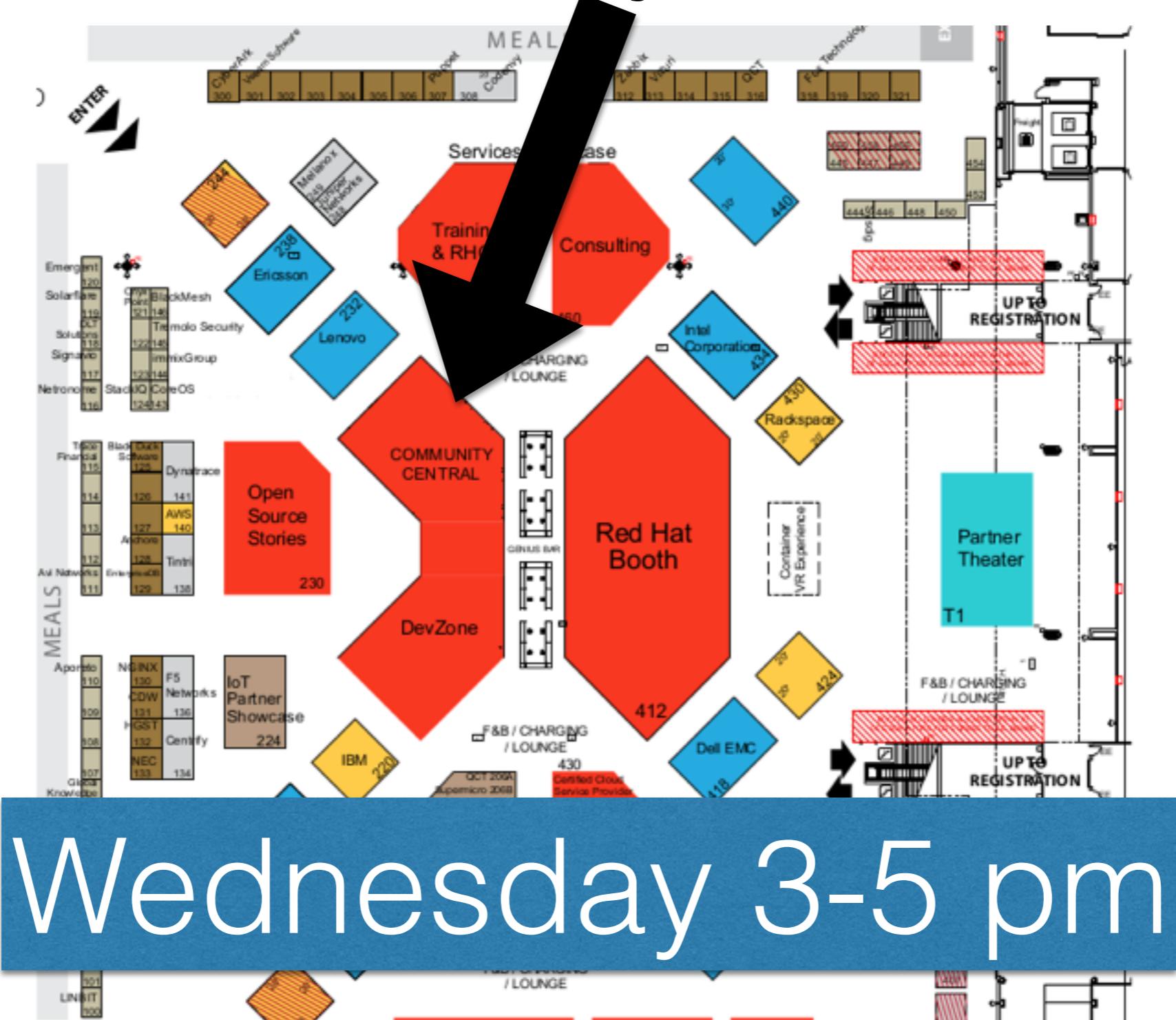
Christian
Posta

Boot Session

Community Central



Claus
Ibsen



Links



@davsclaus



davsclaus

davsclaus.com

- MiniShift
 - <https://www.openshift.org/minishift>
 - fabric8 Maven Tool
 - <https://maven.fabric8.io>
 - Slides and demo source code
 - <https://github.com/davsclaus/minishift-hello>