



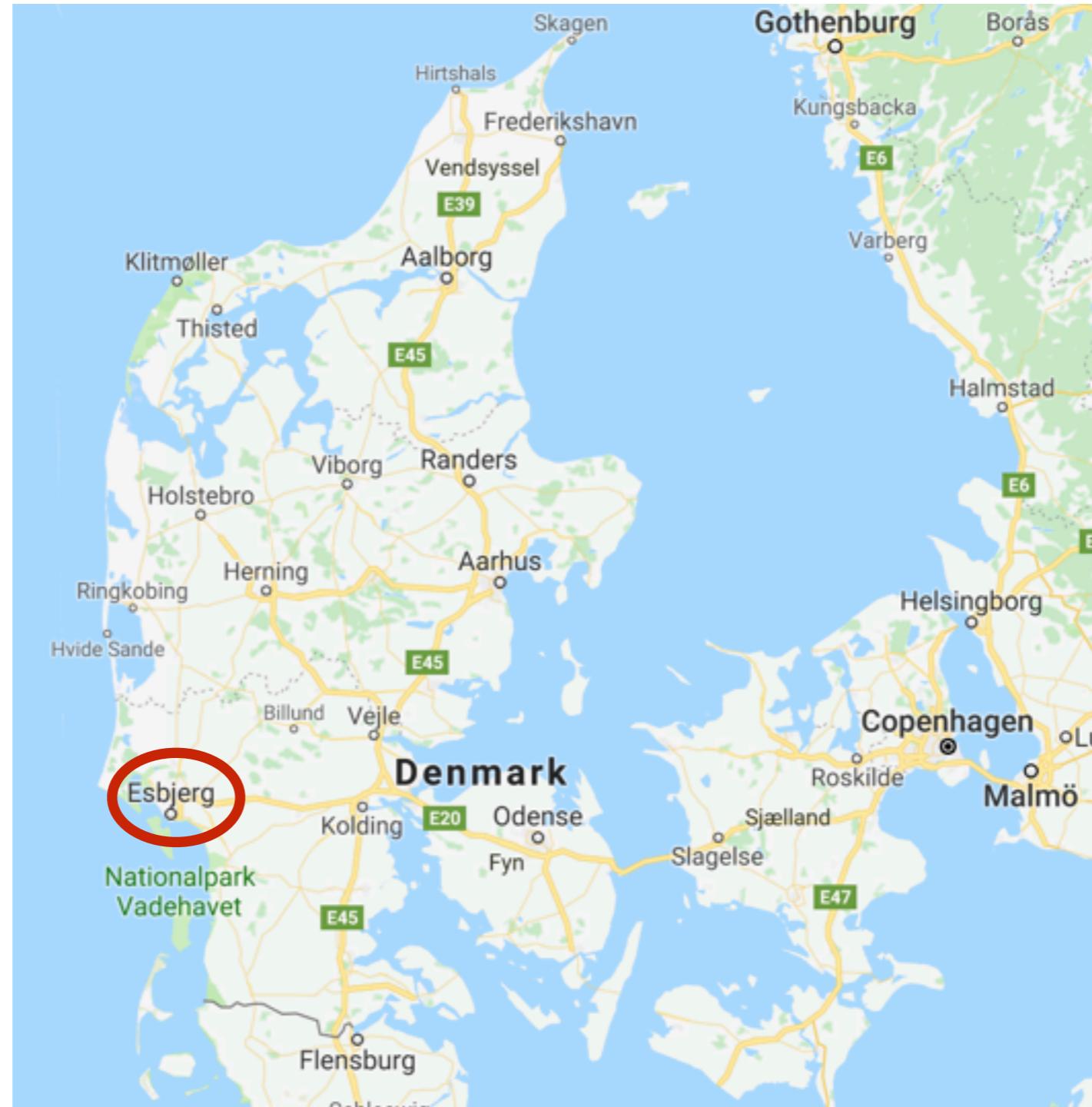
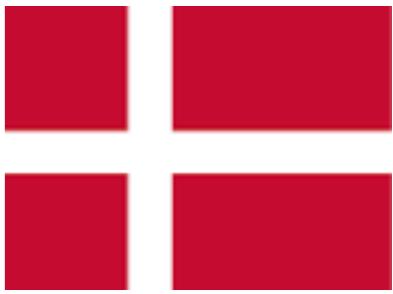
Developing Apache Camel microservices on Kubernetes

Claus Ibsen

 @davsclaus

Boston
February 2018

I'm from Denmark



Walking the city park



Meet Washington



And this "strange Camel"



Claus Ibsen



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



@davsclaus



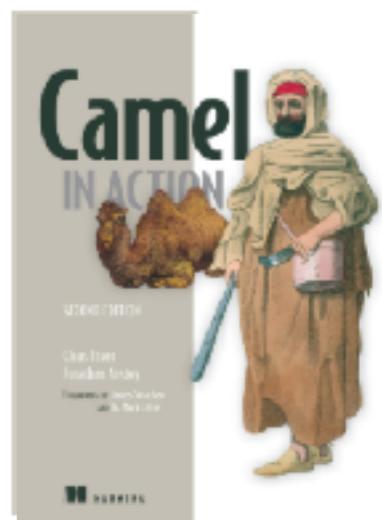
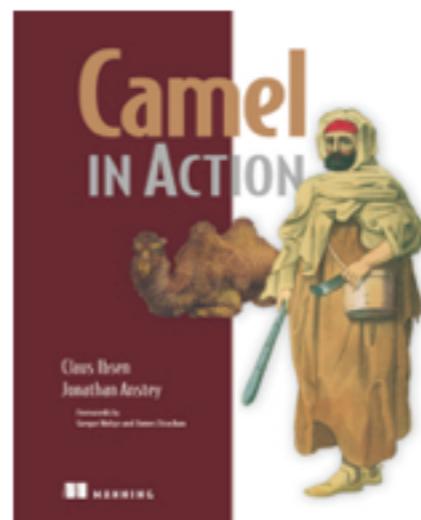
davsclaus



www.davsclaus.com



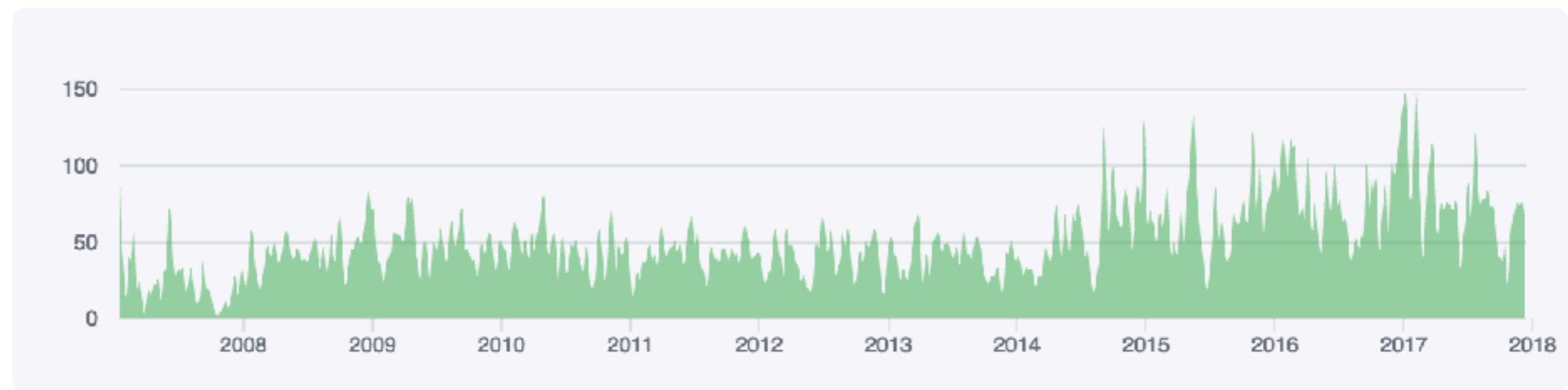
cibsen@redhat.com



Mar 18, 2007 – Feb 15, 2018

Contributions: Commits ▾

Contributions to master, excluding merge commits



davsclaus

13,372 commits 1,607,528 ++ 980,591 --

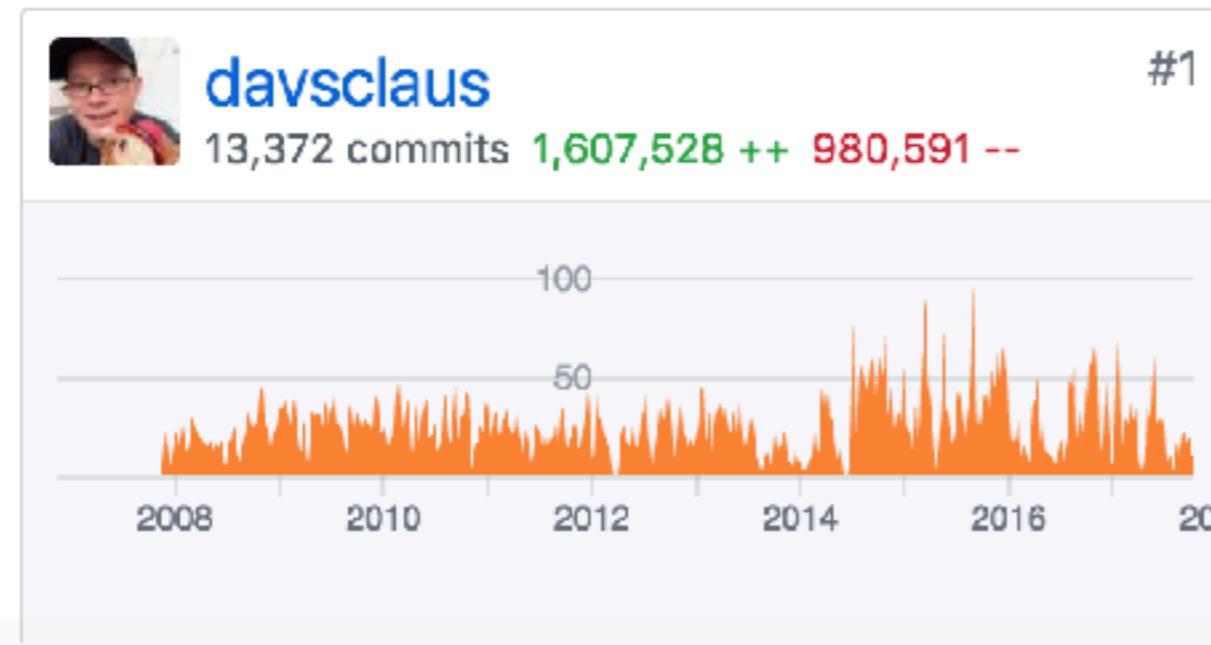
#1



WillemJiang

4,075 commits 389,336 ++ 132,557 --

#2



THE THREE PILLARS OF AGILE INTEGRATION

Key foundational capabilities needed by today's enterprises

DISTRIBUTED INTEGRATION

- LIGHTWEIGHT
- PATTERN BASED
- EVENT ORIENTED
- COMMUNITY SOURCED

FLEXIBILITY

CONTAINERS

- CLOUD NATIVE SOLUTIONS
- LEARN ARTIFACTS, INDIVIDUALLY DEPLOYABLE
- CONTAINER BASED SCALING AND HIGH AVAILABILITY

SCALABILITY

APIs

- WELL DEFINED, REUSABLE, AND WELL MANAGED END-POINTS
- ECOSYSTEM LEVERAGE

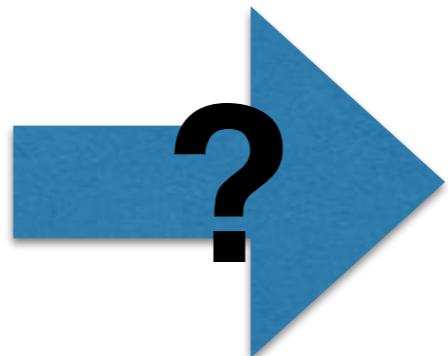
RE-USABILITY

First steps

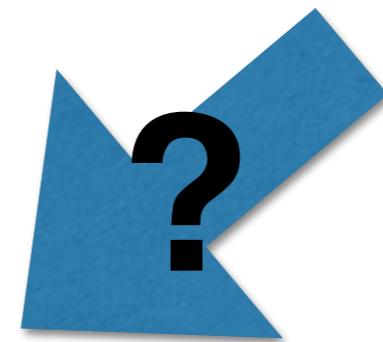
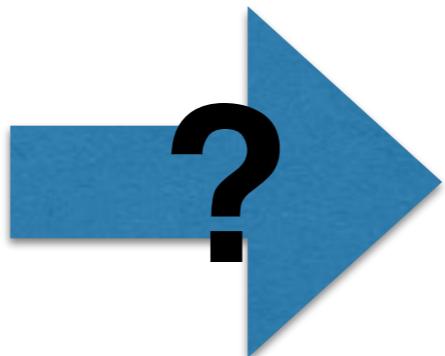
First steps



First steps



First steps



Tip of Iceberg



Running Kubernetes



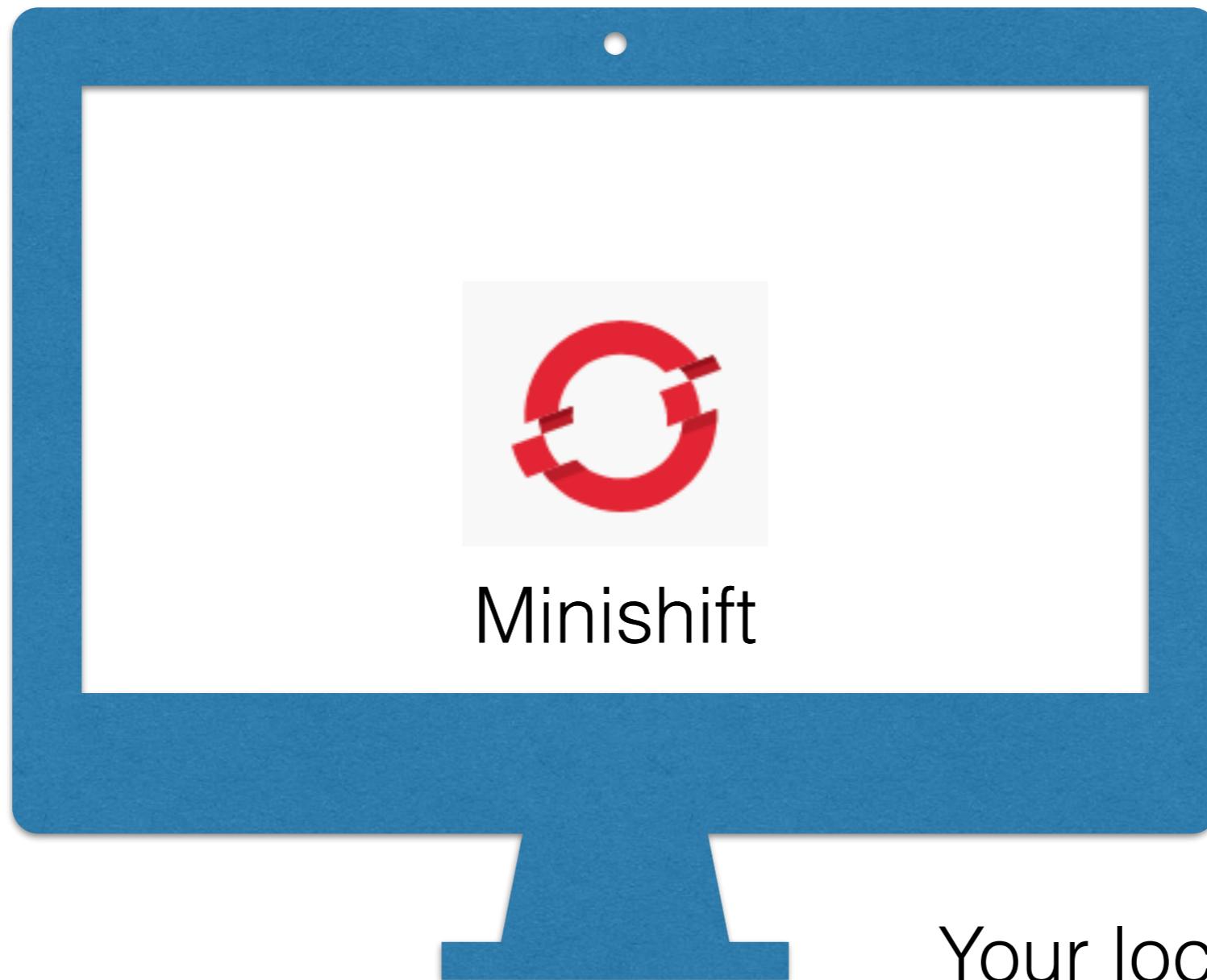
Big "scary" cluster software

Running Local Kubernetes



Minishift

Running Local Kubernetes



Installing Minishift

1. Download Minishift

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

2. Start Minishift

```
$ minishift start
```

<https://docs.openshift.org/latest/minishift/getting-started>

How I installed Minishift

```
davsclaus:/Users/davsclaus/$ minishift start
Starting local OpenShift cluster using 'xhyve' hypervisor...
Downloading ISO 'https://github.com/minishift/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
```

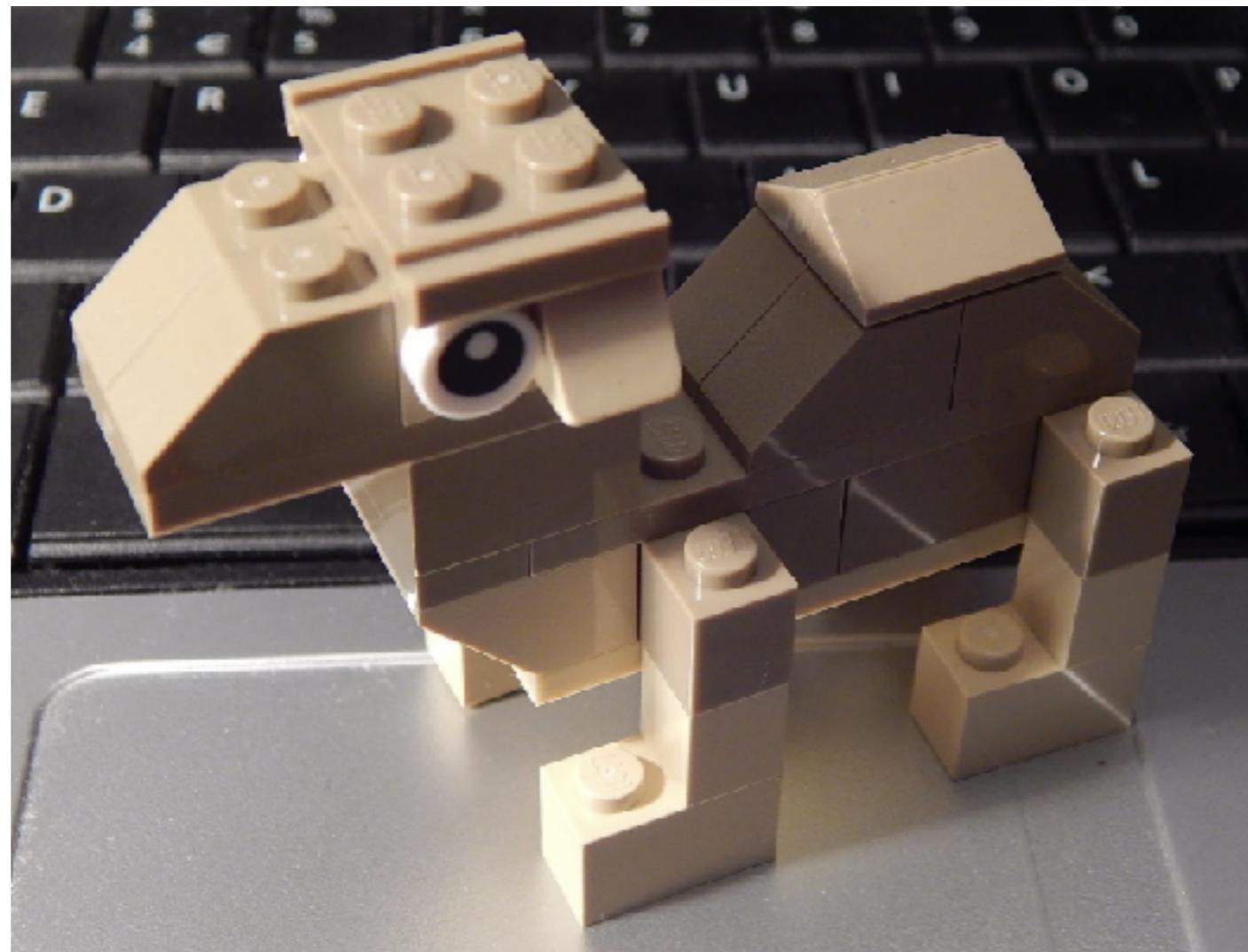
Web Console

```
$ minishift console
```

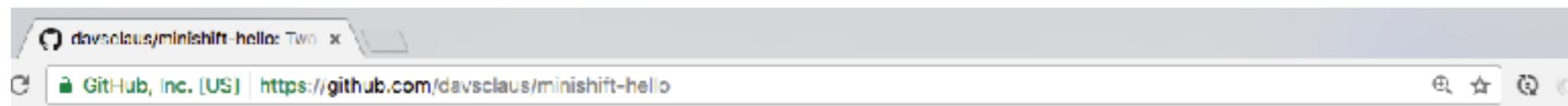
The screenshot shows the OpenShift Web Console interface. At the top, there's a navigation bar with a red, yellow, and green status indicator, the title "OpenShift Web Console", and a "Not Secure" warning in the address bar. The address bar also displays the URL <https://192.168.64.4:8443/console/project/myproject/overview>. Below the header, the main menu includes a "Project" dropdown set to "My Project", a "Home" icon, "Add to Project" button, and user profile "developer". On the left, a sidebar lists "Overview", "Applications", "Builds", "Resources", "Storage", and "Monitoring". The "Overview" tab is selected. The main content area is titled "Deployments" and lists three entries:

Deployment Name	Status	Pod Count	More Options
client, #12	Green circle	1 pod	⋮
client-hystrix, #4	Grey circle	0 pods	⋮
heloswarm, #4	Green circle	1 pod	⋮

Build a Camel Demo Time



Source Code & Slides



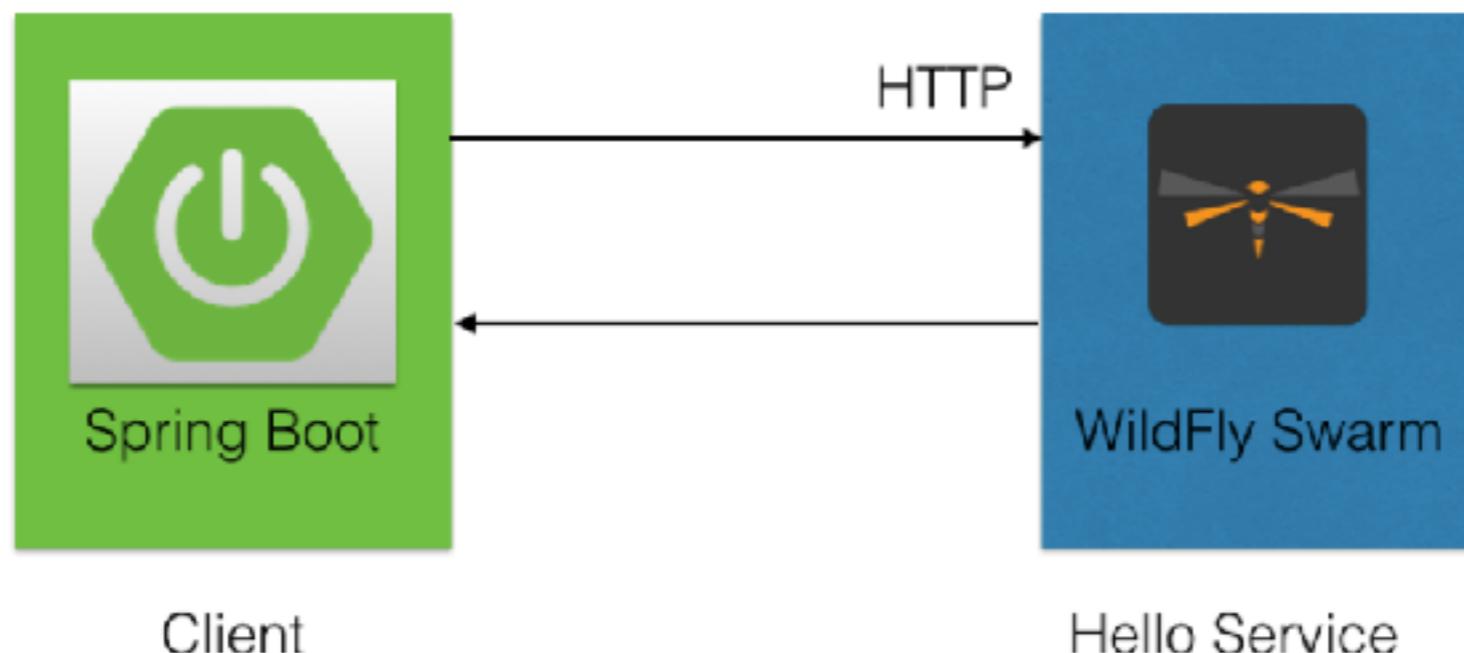
minishift-hello

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:



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

Hello Service

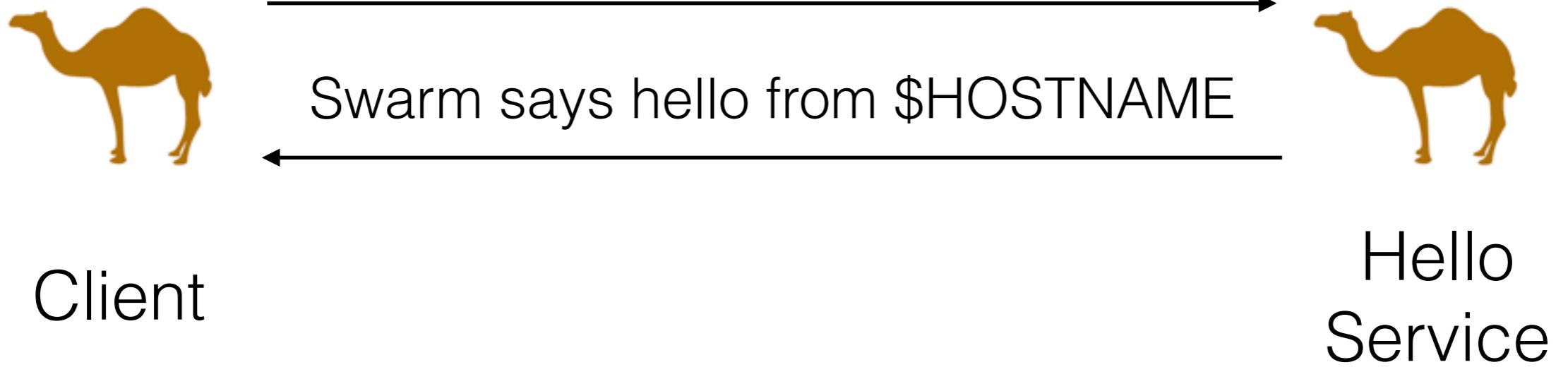


Client

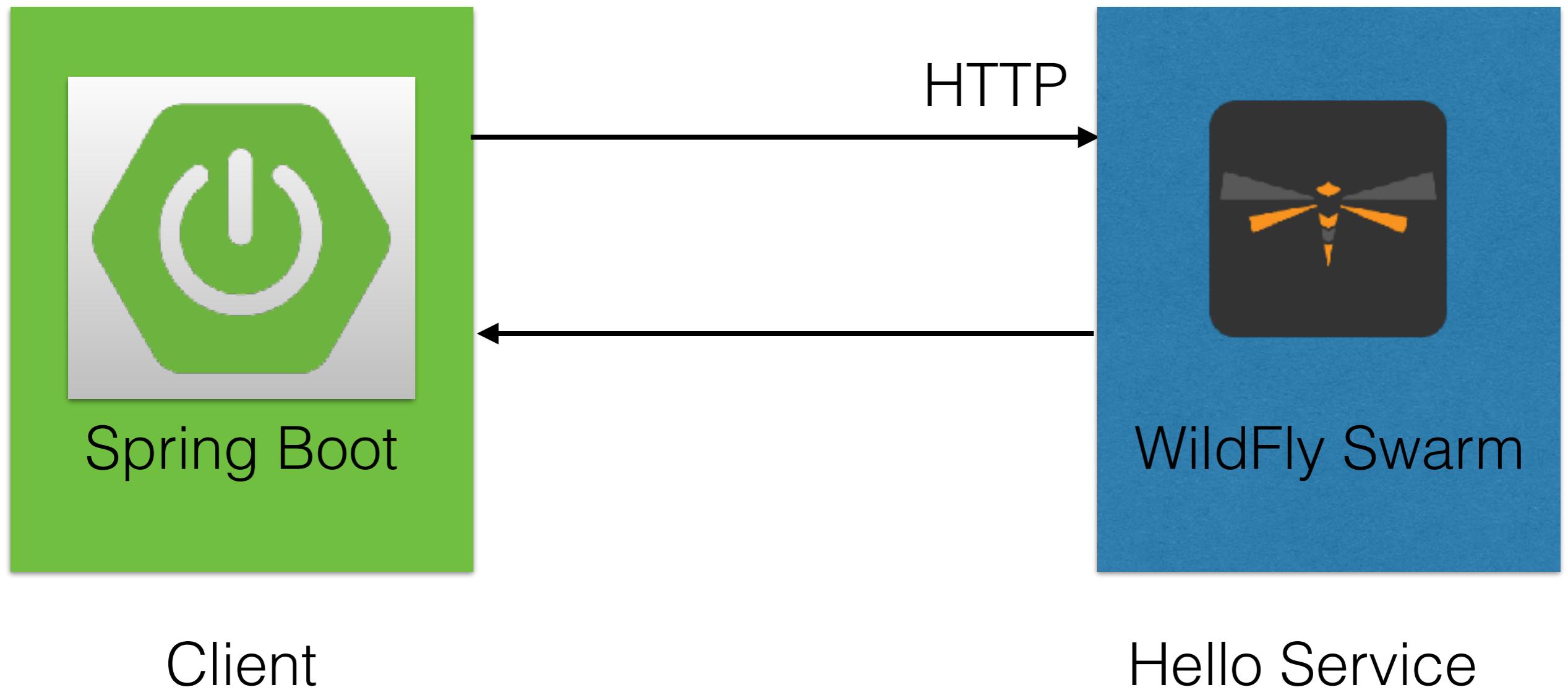


Hello
Service

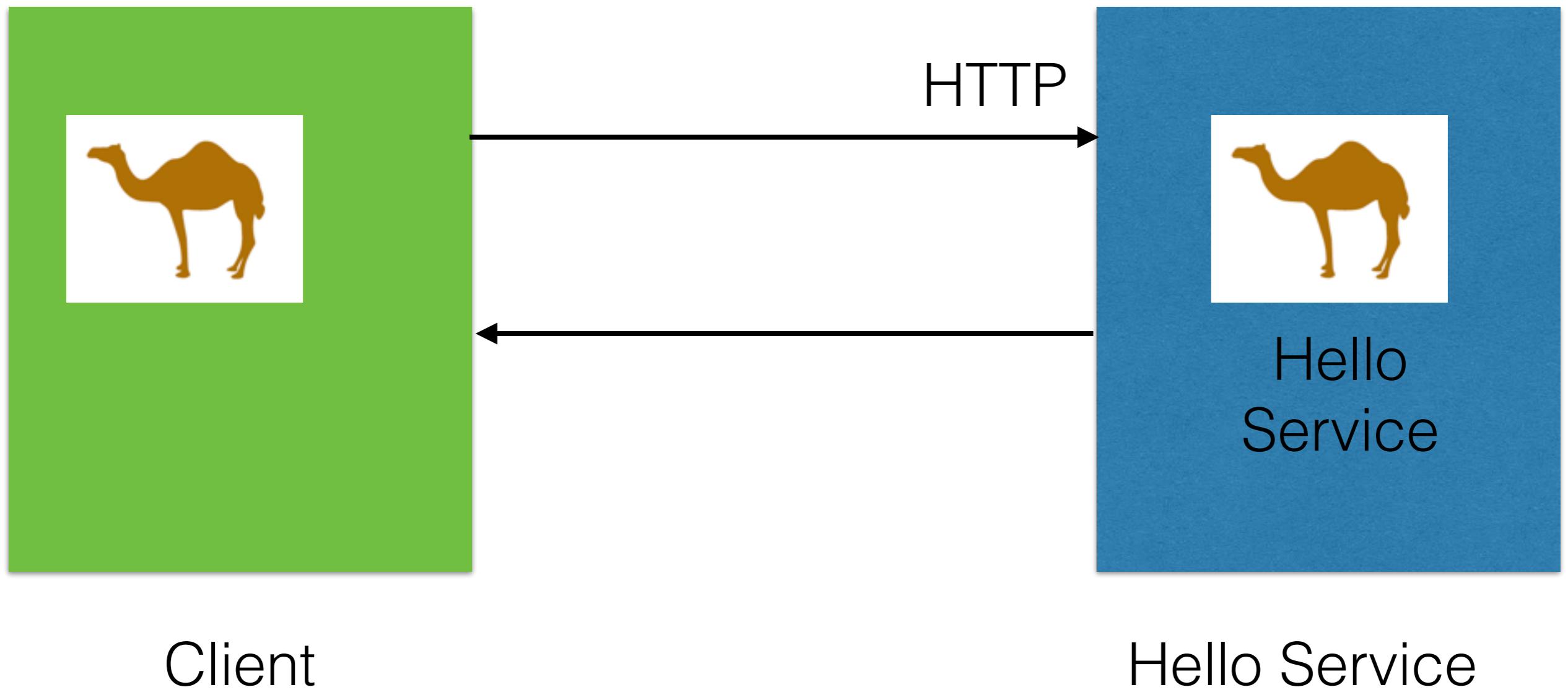
Hello Service



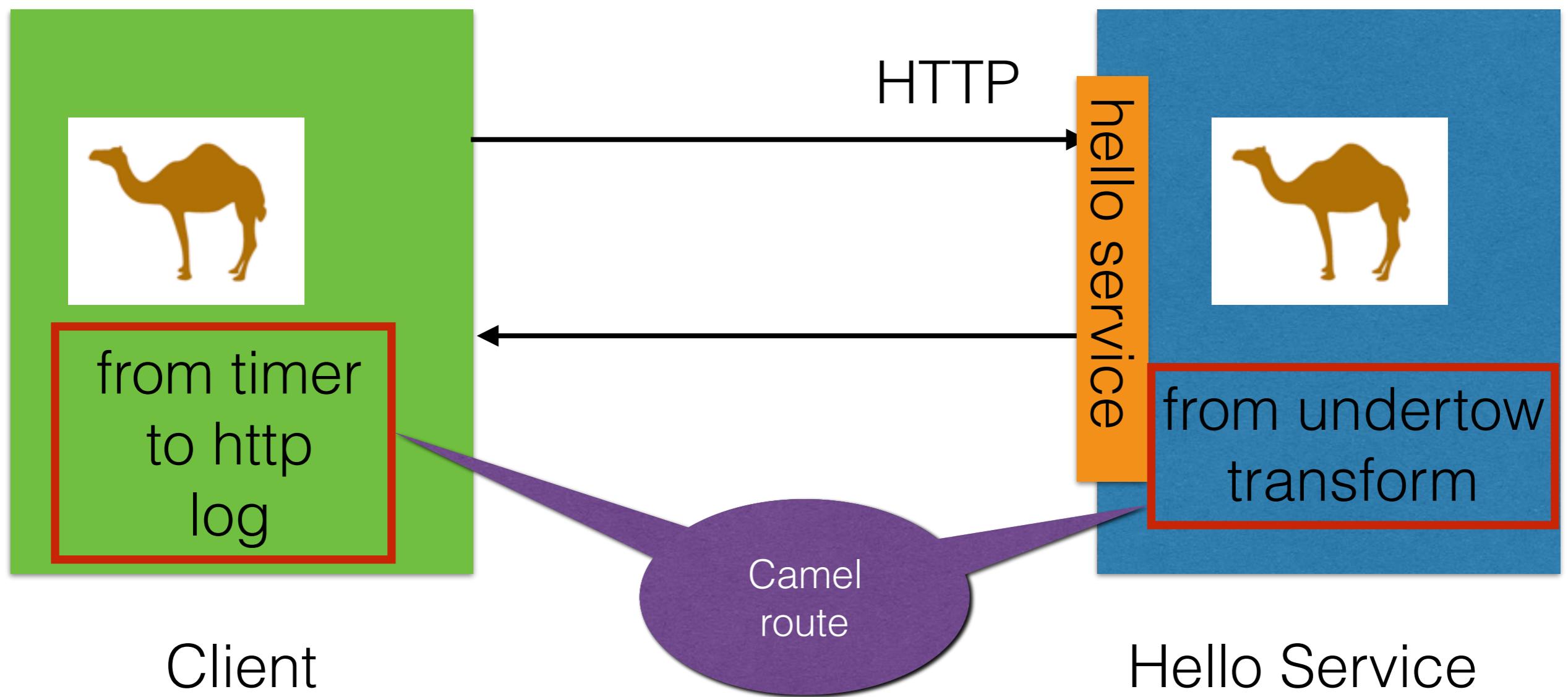
Implementation



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 : **All**

Selected dependencies
Camel CDI **Camel Undertow**



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 displaying the Spring Initializr website at start.spring.io. The title bar shows the browser window is titled "Spring Initializr". The address bar shows the URL "start.spring.io". The page itself has a dark header with the text "SPRING INITIALIZR bootstrap your application now". Below the header, there's a form to "Generate a Maven Project with Spring Boot 1.5.3". The "Project Metadata" section includes fields for "Artifact coordinates" (Group: com.foo, Artifact: client). The "Dependencies" section includes a search bar with "Web, Security, JPA, Actuator, Devtools..." listed, and a "Selected Dependencies" section with "Web X", "Actuator X", and "Apache Camel X". A large green "Generate Project" button is at the bottom.

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

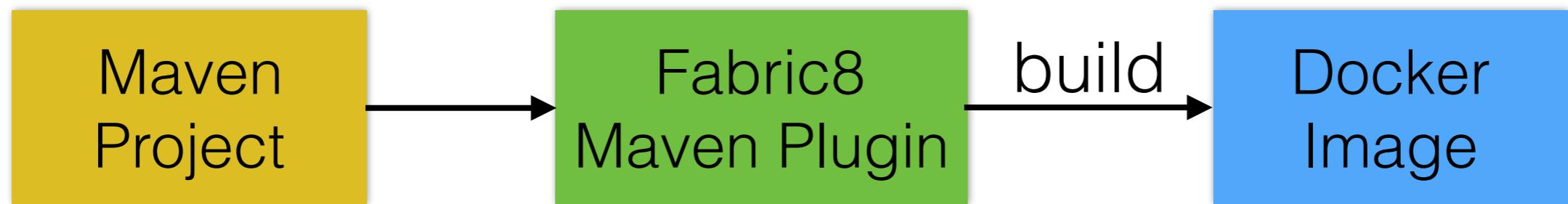
```
@Component  
public class MyRoute extends RouteBuilder {  
  
    @Override  
    public void configure() throws Exception {  
        from( uri: "timer:foo?period=2000")  
            .to("http4:localhost:8080/hello")  
            .log("${body}");  
    }  
}
```

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>3.5.33</version>
</plugin>
```

<https://maven.fabric8.io>

Setup Shell

- Prepare command shell

```
$ minikube docker-env  
eval $(minikube docker-env)
```

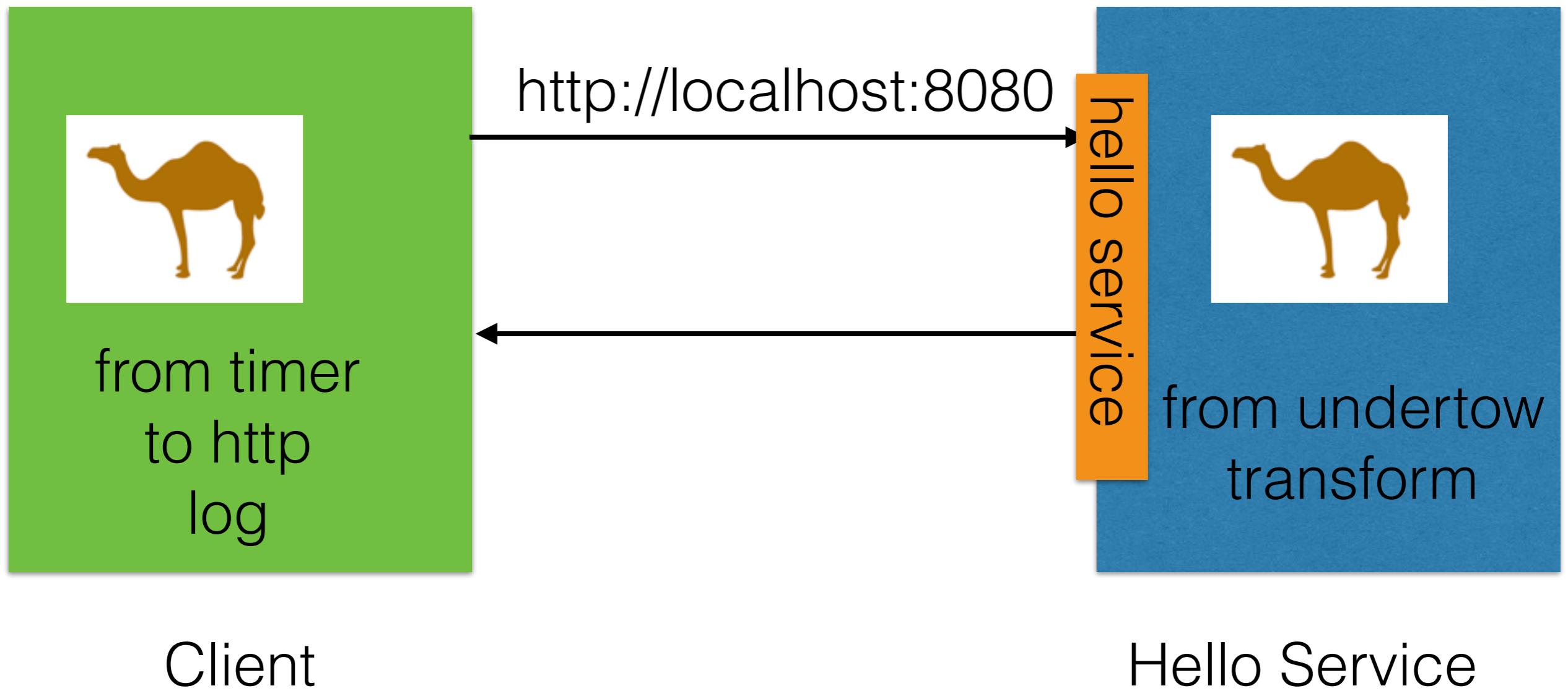
```
davsclaus:/Users/davsclaus/webinar/helloswarm (master)$ minikube docker-env  
export DOCKER_TLS_VERIFY="1"  
export DOCKER_HOST="tcp://192.168.64.5:2376"  
export DOCKER_CERT_PATH="/Users/davsclaus/.minikube/certs"  
export DOCKER_API_VERSION="1.23"  
# Run this command to configure your shell:  
# eval $(minikube docker-env)  
davsclaus:/Users/davsclaus/webinar/helloswarm (master)$ eval $(minikube docker-env)
```

Build Docker Image

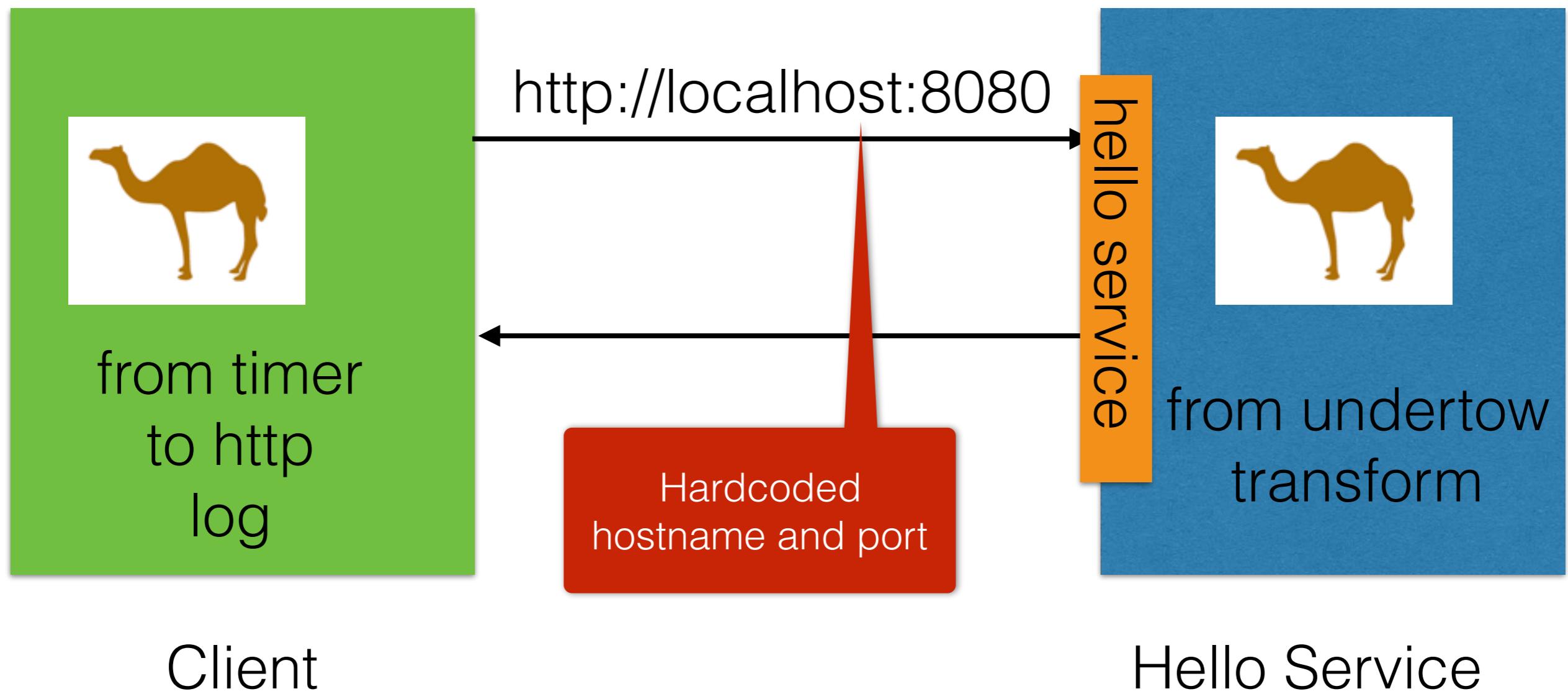
```
[INFO] --- fabric8-maven-plugin:3.3.5:resource (fmp) @ helloswarm ---
[INFO] F8: Running in Kubernetes mode
[INFO] F8: Running generator wildfly-swarm
[INFO] F8: wildfly-swarm: Using Docker image fabric8/java-jboss-openjdk8-jdk:1.2 as base / builder
[INFO] F8: fmp-controller: Adding a default Deployment
[WARNING] F8: fmp-service: Implicit service port mapping to port 80 has been disabled for the used
either use set the config port = 80 or use legacyPortMapping = true. See https://maven.fabric8.io/
[INFO] F8: fmp-service: Adding a default service 'helloswarm' with ports [8080]
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ helloswarm ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to /Users/davsclaus/Documents/workspace/minishift-hello/helloswarm/
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ helloswarm ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/davsclaus/Documents/workspace/minishift-hello/hel
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ helloswarm ---
[INFO] No sources to compile
```

mvn fabric8:build

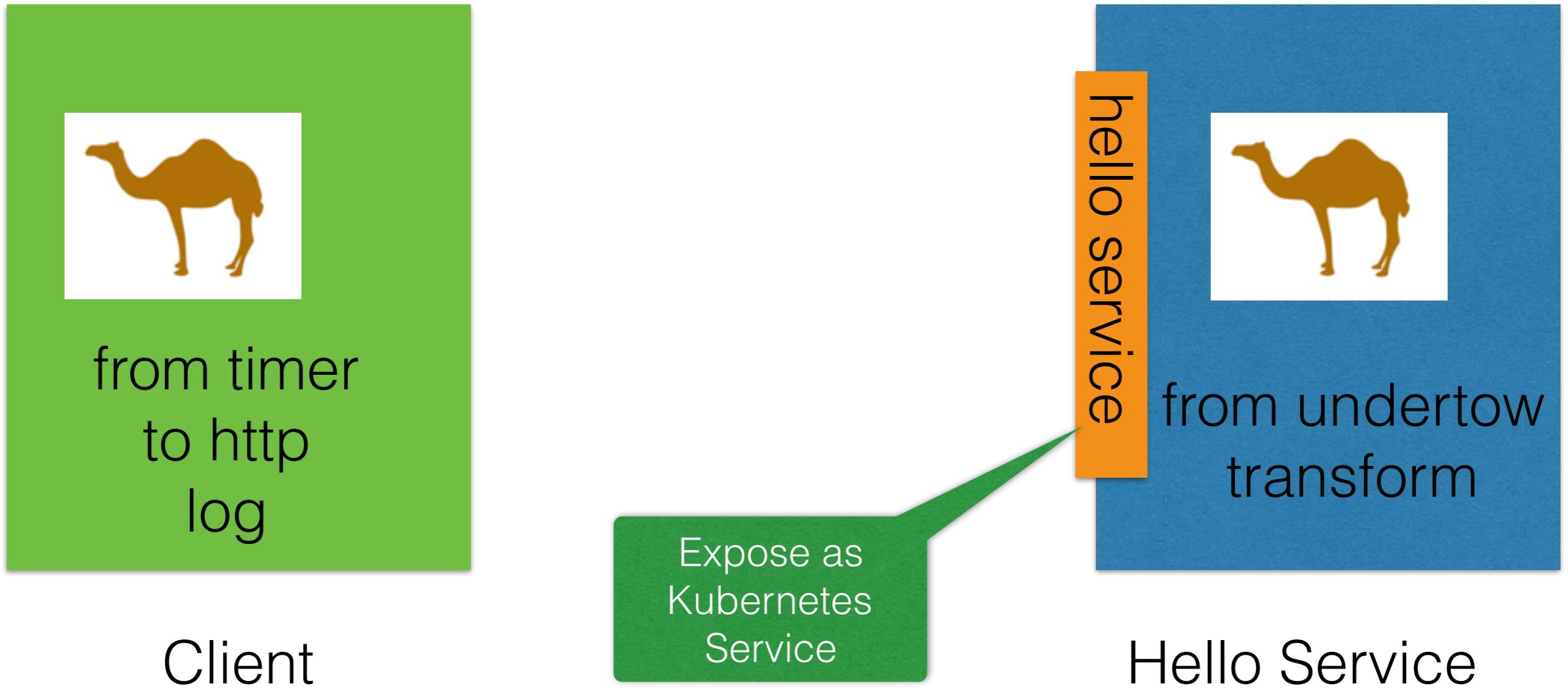
Our Demo



Static vs Dynamic Platform



Dynamic Platform



Dynamic Platform

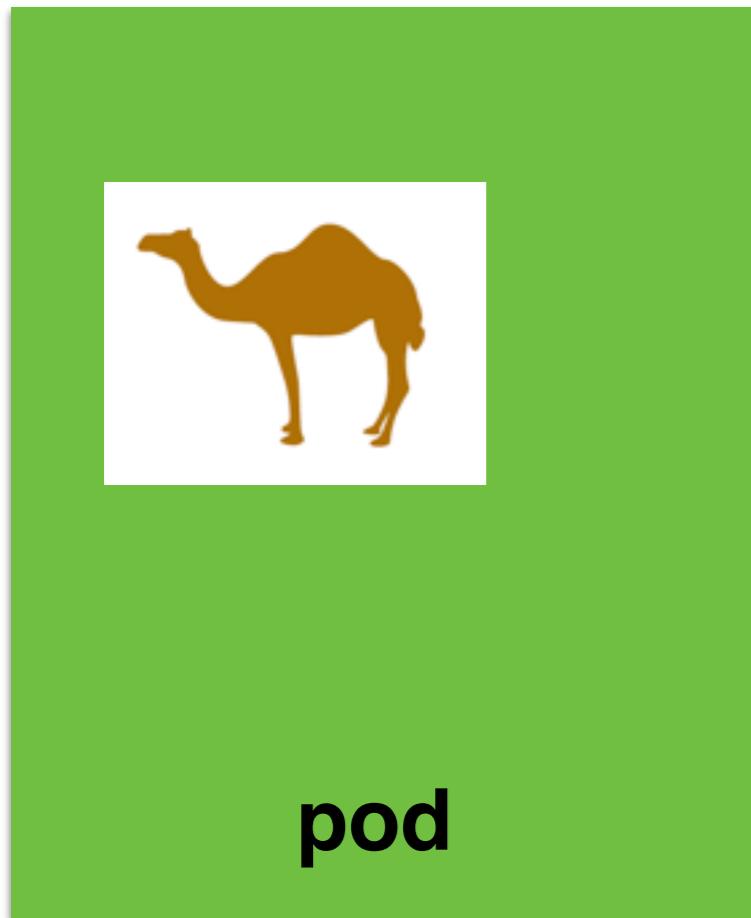


Client



Hello Service

Dynamic Platform



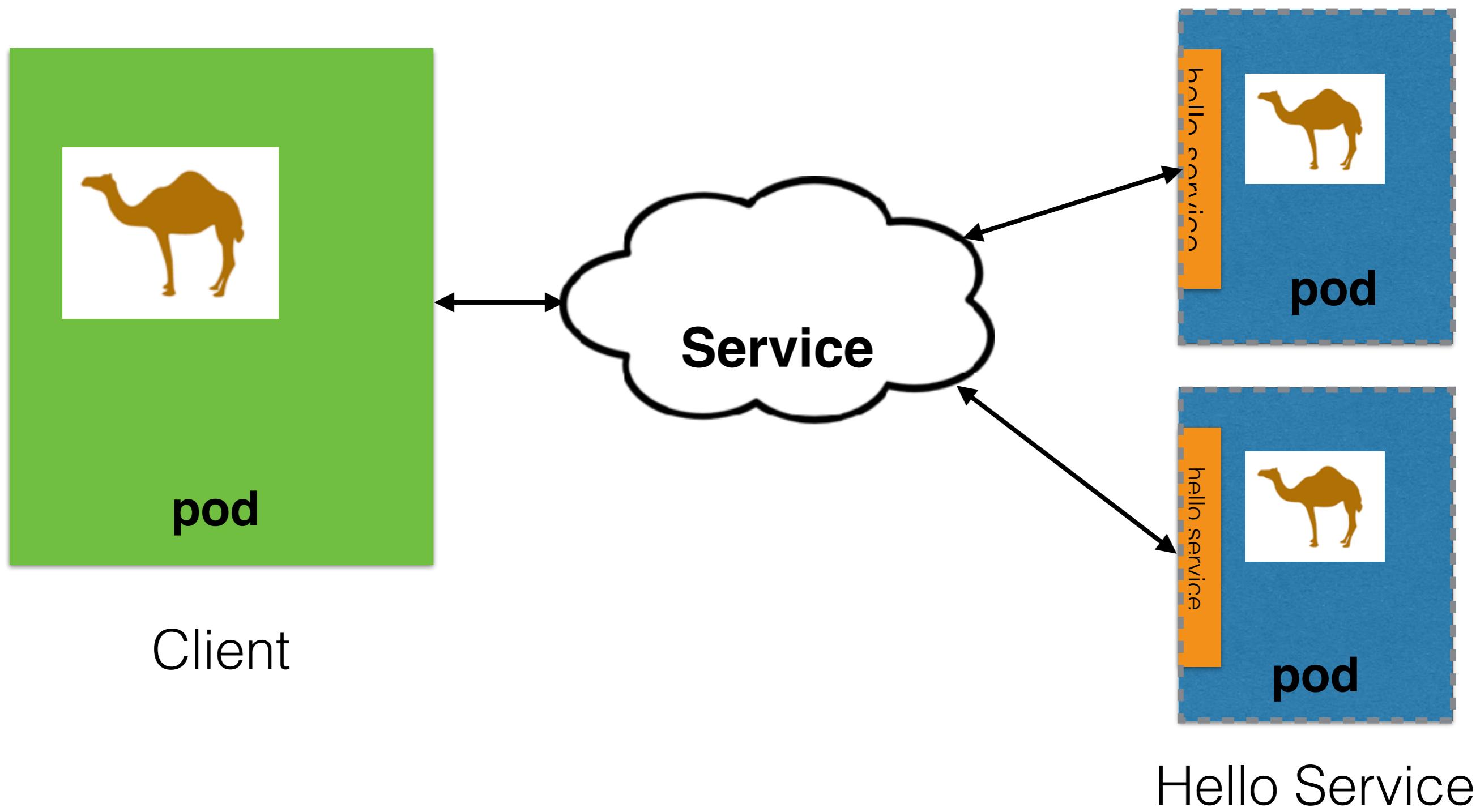
Client



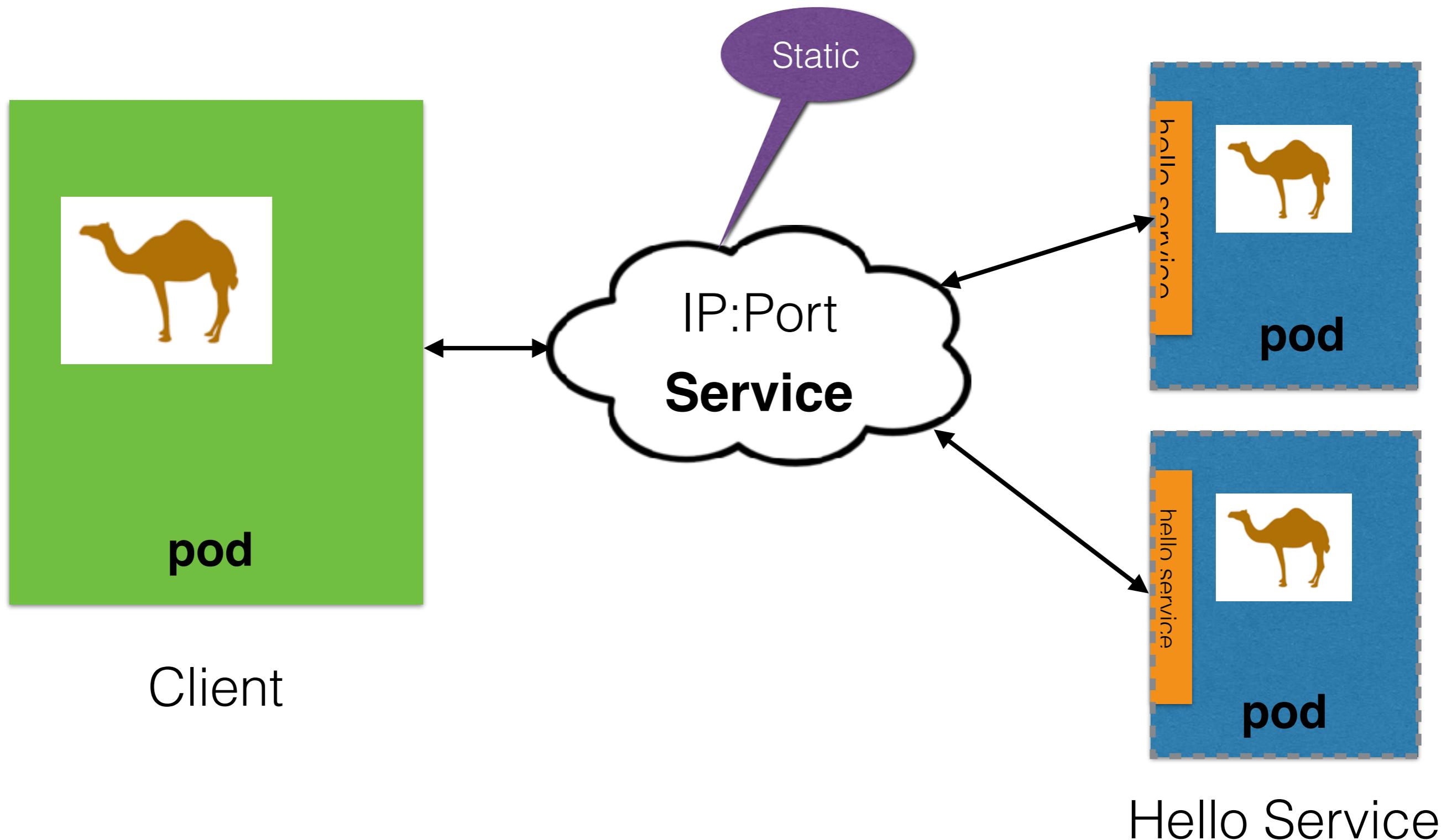
Hello Service



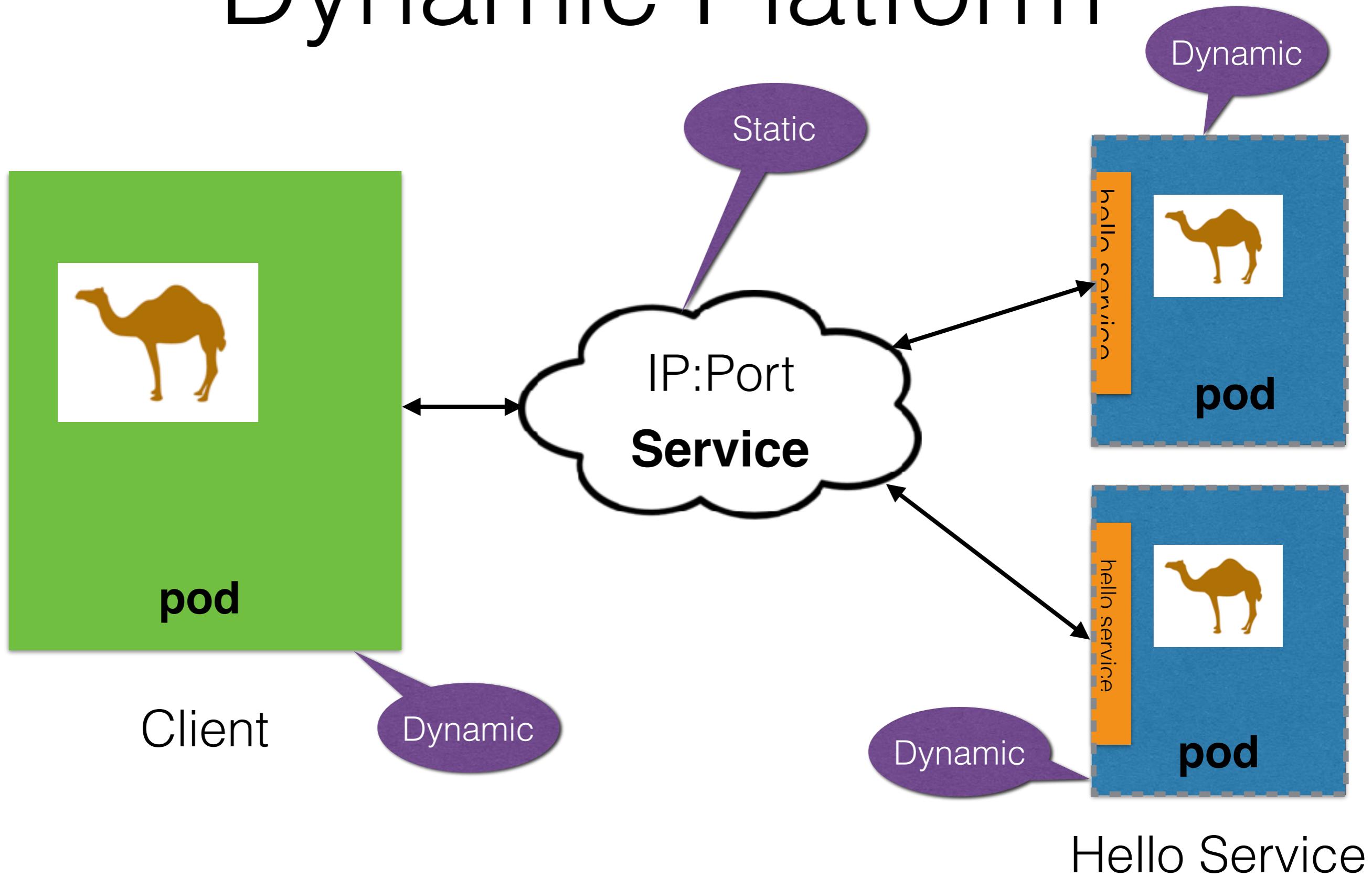
Dynamic Platform



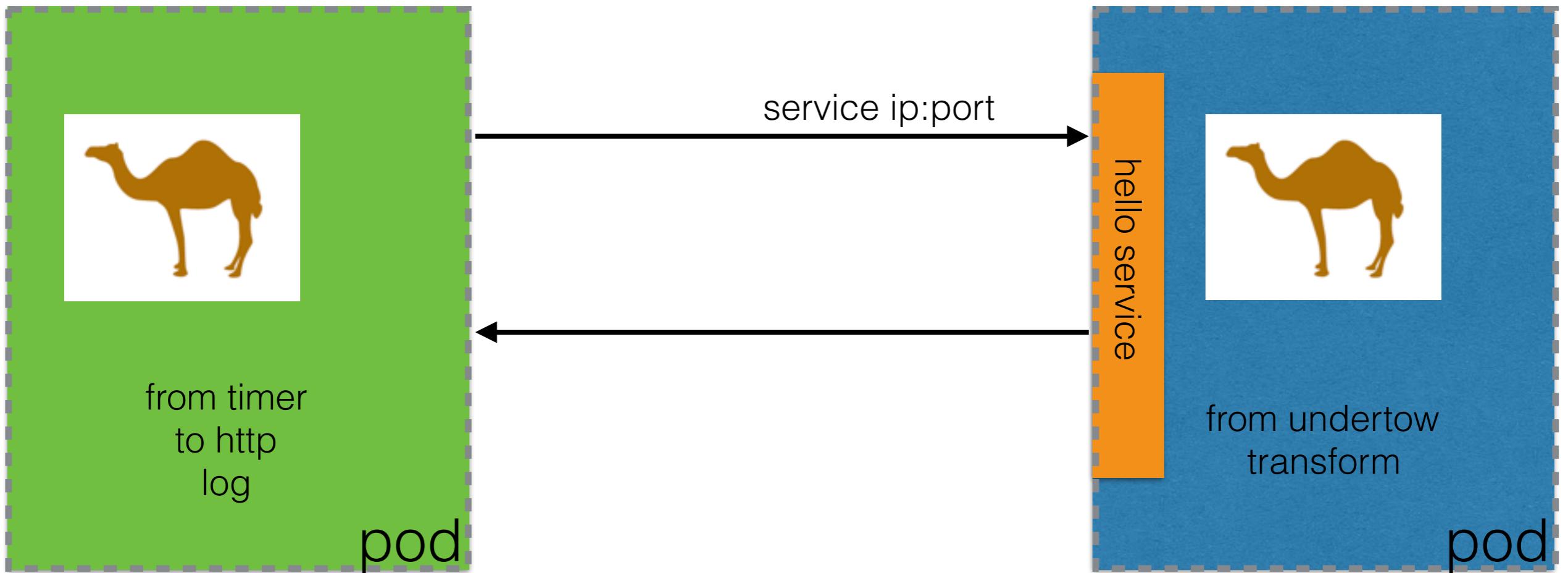
Dynamic Platform



Dynamic Platform



Kubernetes Service from user point of view



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



Service Discovery using DNS is also available

Service using ENV



from timer
to http
log

- {{service:name}}

Client

```
@Component
public class MyRoute extends RouteBuilder {

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

Service using DNS



from timer
to http
log

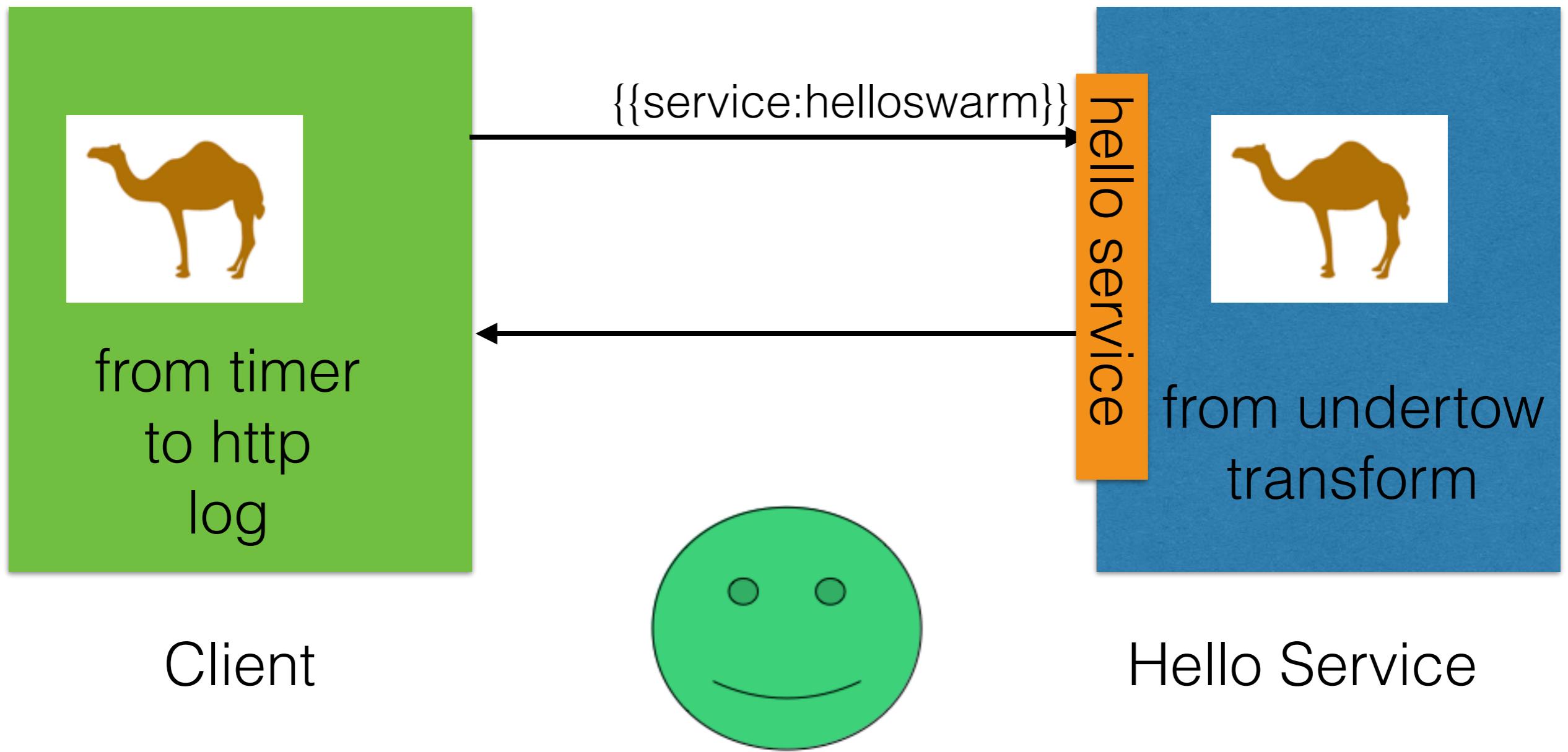
- servicename:port

Client

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from(uri: "timer:too?period=2000")
            .to("http4[helloswarm:8080]/hello")
            .log("${body}");
    }
}
```

Ready to run in Kubernetes

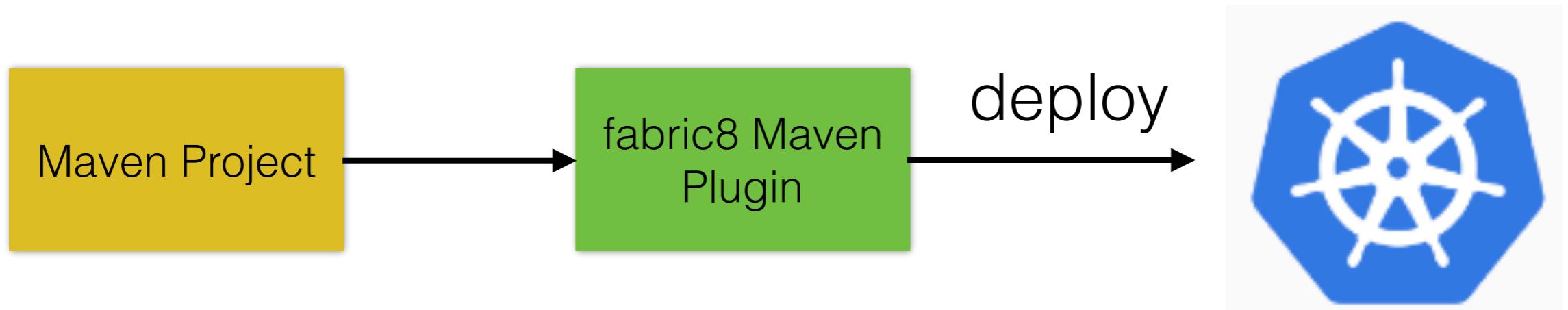


How to deploy to Kubernetes?

Maven Project



How to deploy to Kubernetes?



Deploy - Hello Service

hello service



from undertow
transform

- \$ mvn fabric8:deploy

Hello Service

```
[INFO] --- fabric8-maven-plugin:3.3.5:resource (fmp) @ helloswarm ---
[INFO] F8: Running in Kubernetes mode
[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
either use set the config port = 80 or use legacyPortMapping = true. See https://maven.fabric8.io/
[INFO] F8: fmp-service: Adding a default service 'helloswarm' with ports [8080]
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ helloswarm ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to /Users/davsclaus/Documents/workspace/minishift-hello/helloswarm/
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ helloswarm ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/davsclaus/Documents/workspace/minishift-hello/hel
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ helloswarm ---
[INFO] No sources to compile
```

Deploy - Client



from timer
to http
log

- \$ mvn fabric8:deploy

Client

```
[INFO] >>> fabric8-maven-plugin:3.5.33: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:3.5.33:resource (fmp) @ client ---
[INFO] F8: Running in Kubernetes mode
[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
[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='
[INFO] F8: spring-boot-health-check: Adding liveness probe on port 8080, path='/health', scheme='T
[INFO] F8: fmp-revision-history: Adding revision history limit to 2
[INFO] F8: f8-icon: Adding icon for deployment
[INFO] F8: f8-icon: Adding icon for service
[INFO] F8: validating /Users/davsclaus/Documents/workspace/minishift-hello/client/target/classes/M
tconfig.yml resource
[INFO] F8: validating /Users/davsclaus/Documents/workspace/minishift-hello/client/target/classes/M
```

Run - Client



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

- \$ mvn fabric8:run

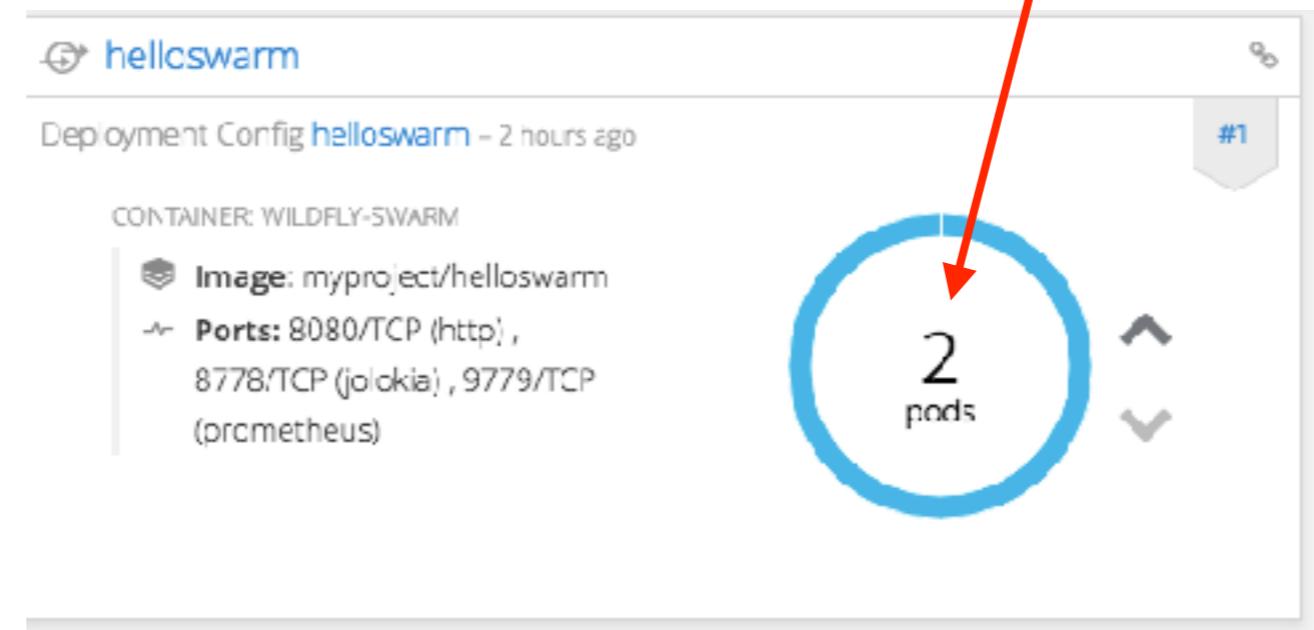
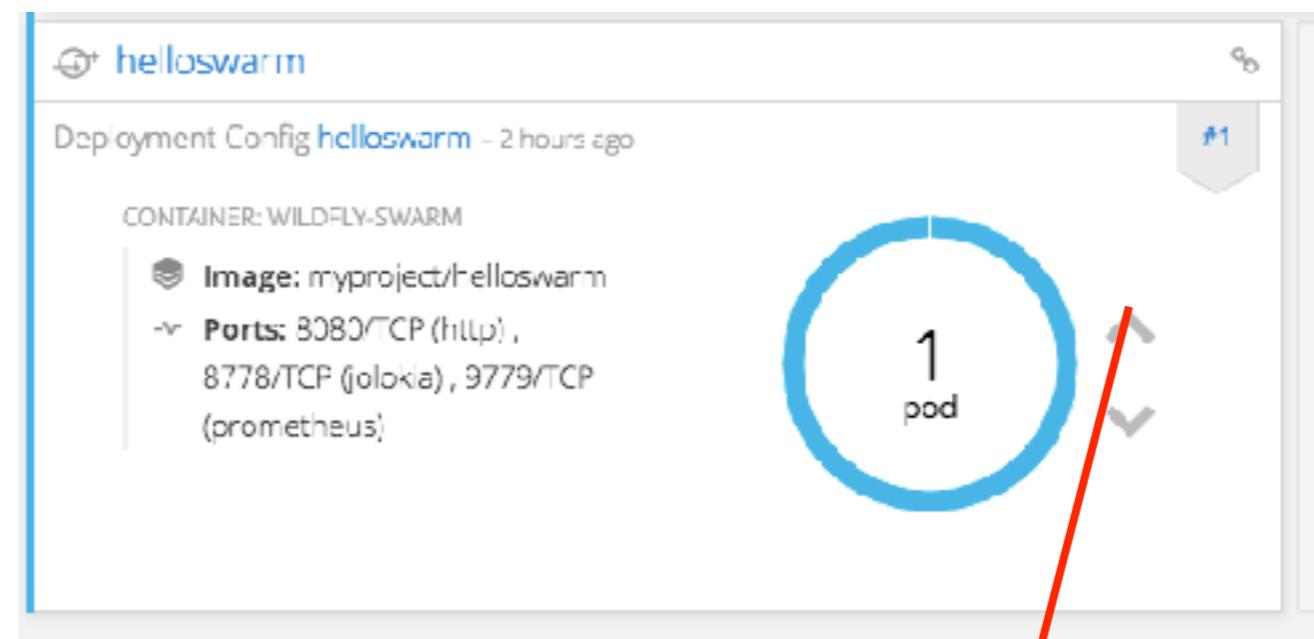
Client

Ready to run in Kubernetes



Scaling

- Change deployment replicas



Load balancing
is random

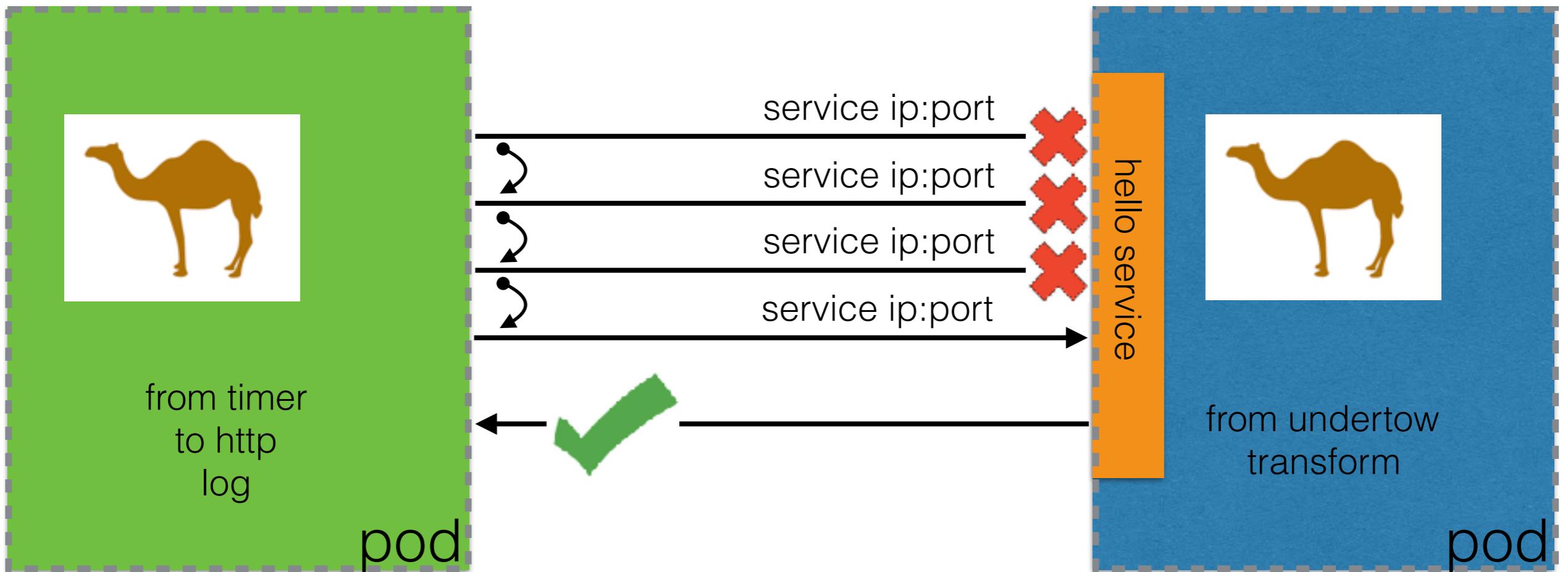
Scaling

- 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

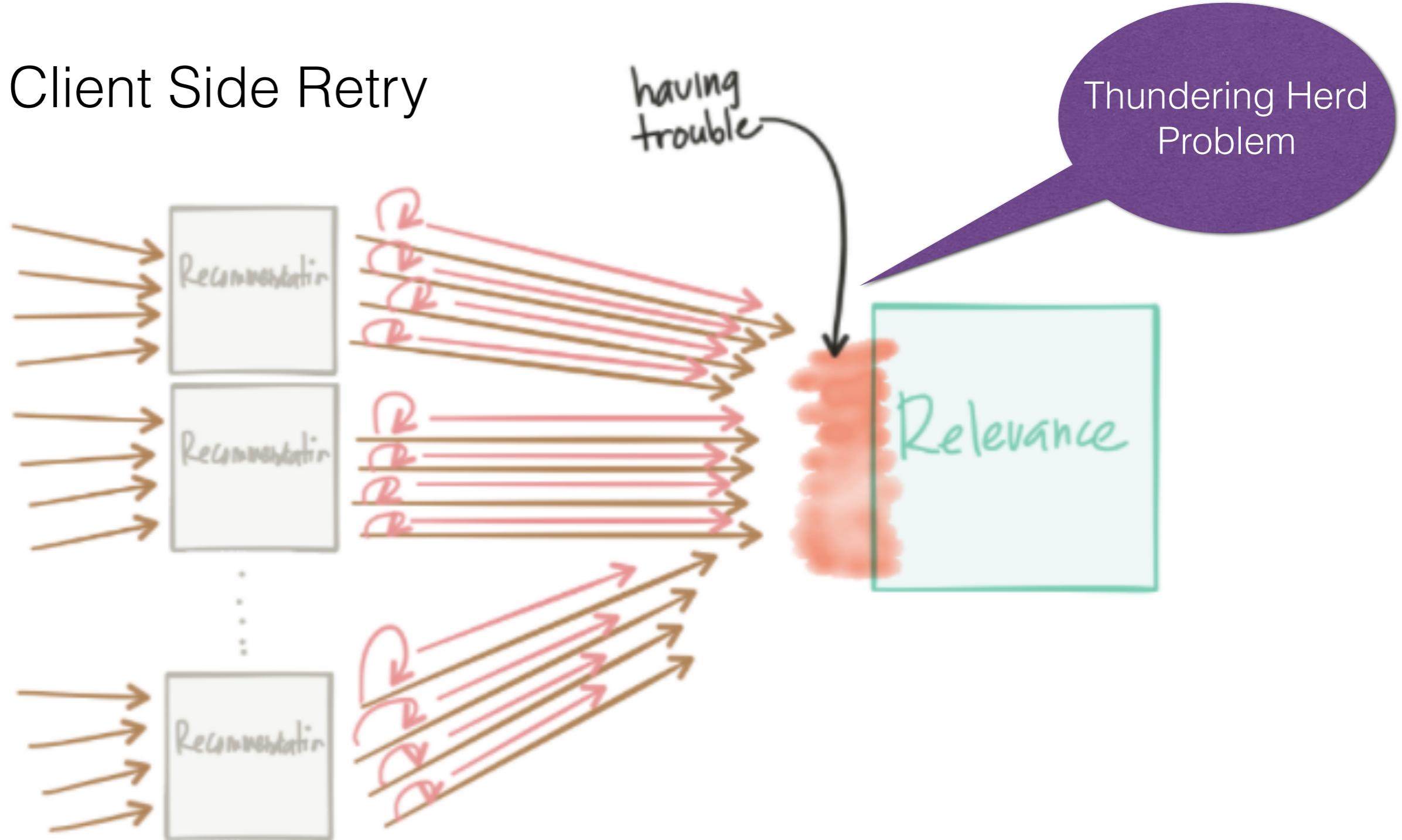
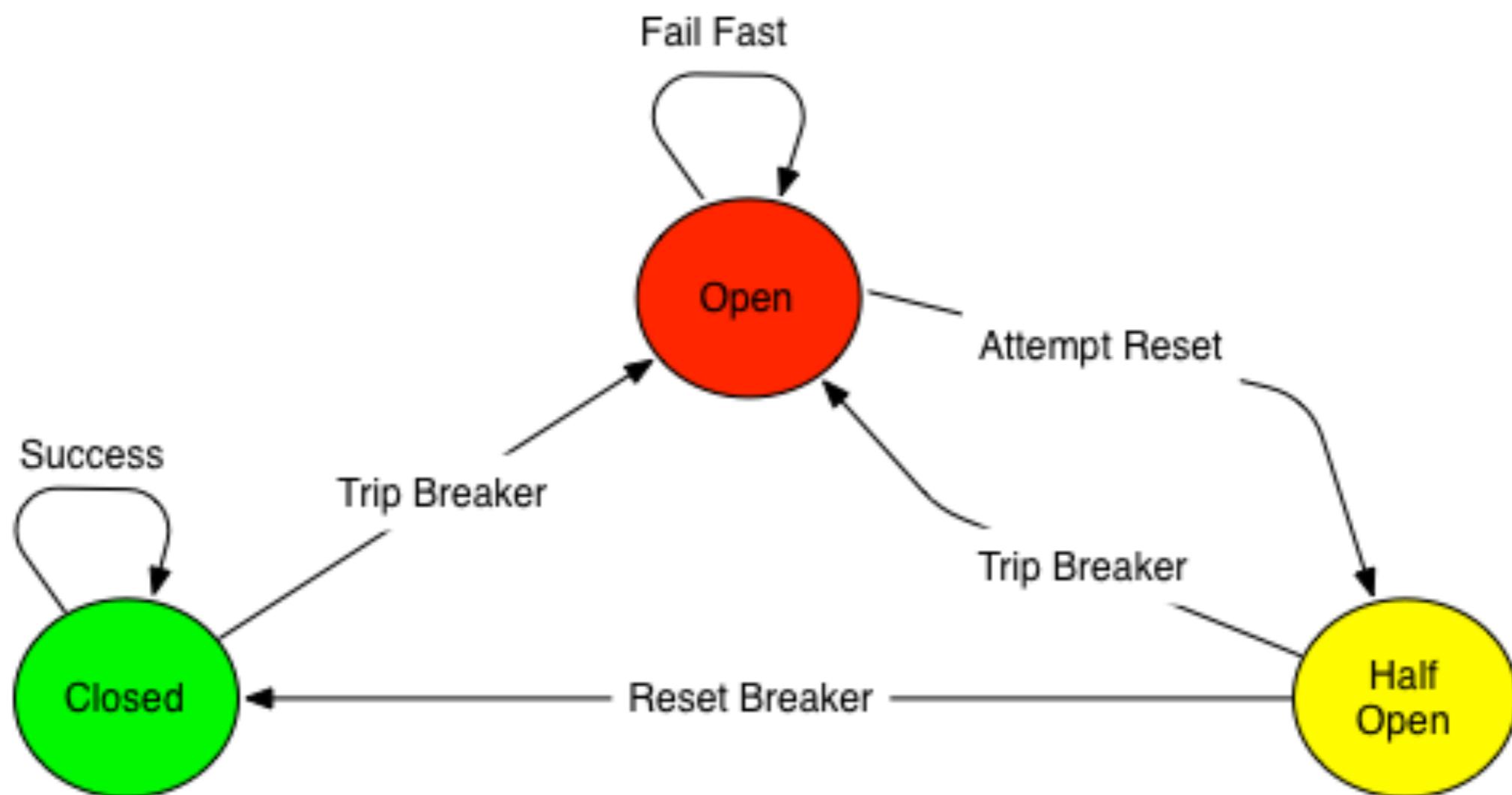


Figure by Christian Posta

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}");
    }
}
```

Hystrix Dashboard

Hystrix Stream: undefined

Circuit

Sort: [Error then Volume](#) | [Alphabetical](#) | [Volume](#) | [Error](#) | [Mean](#) | [Median](#) | [90](#) | [99](#) | [99.5](#)



Host: 0.5/s

Cluster: 0.5/s

Circuit [Closed](#)

Hosts	1	90th	31ms
Median	19ms	99th	31ms
Mean	18ms	99.5th	31ms

Thread Pools

Sort: [Alphabetical](#) | [Volume](#) |



Instructions in readme how to
install and use

Hystrix Demo



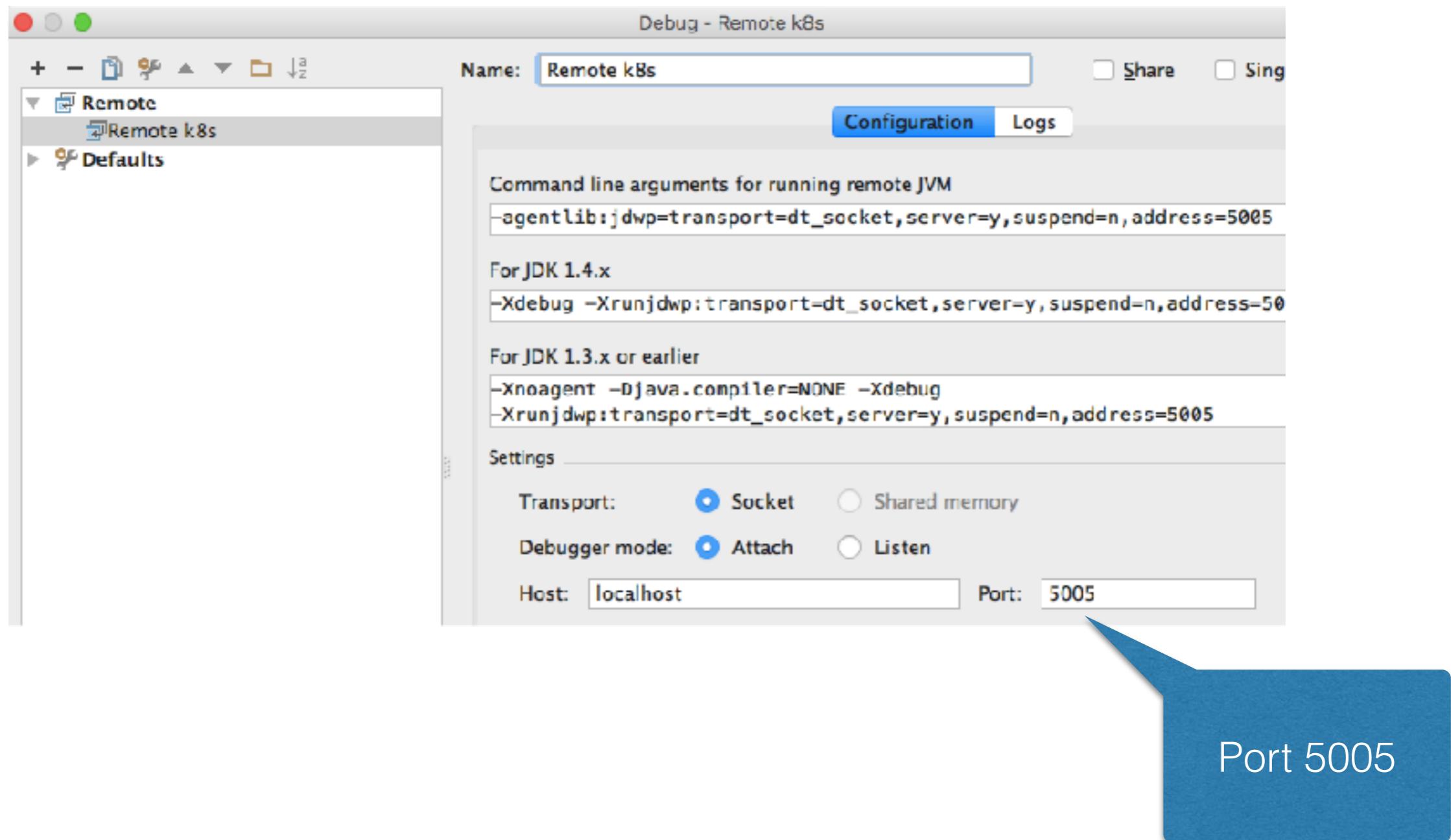
Debugging

- Debugging Pods

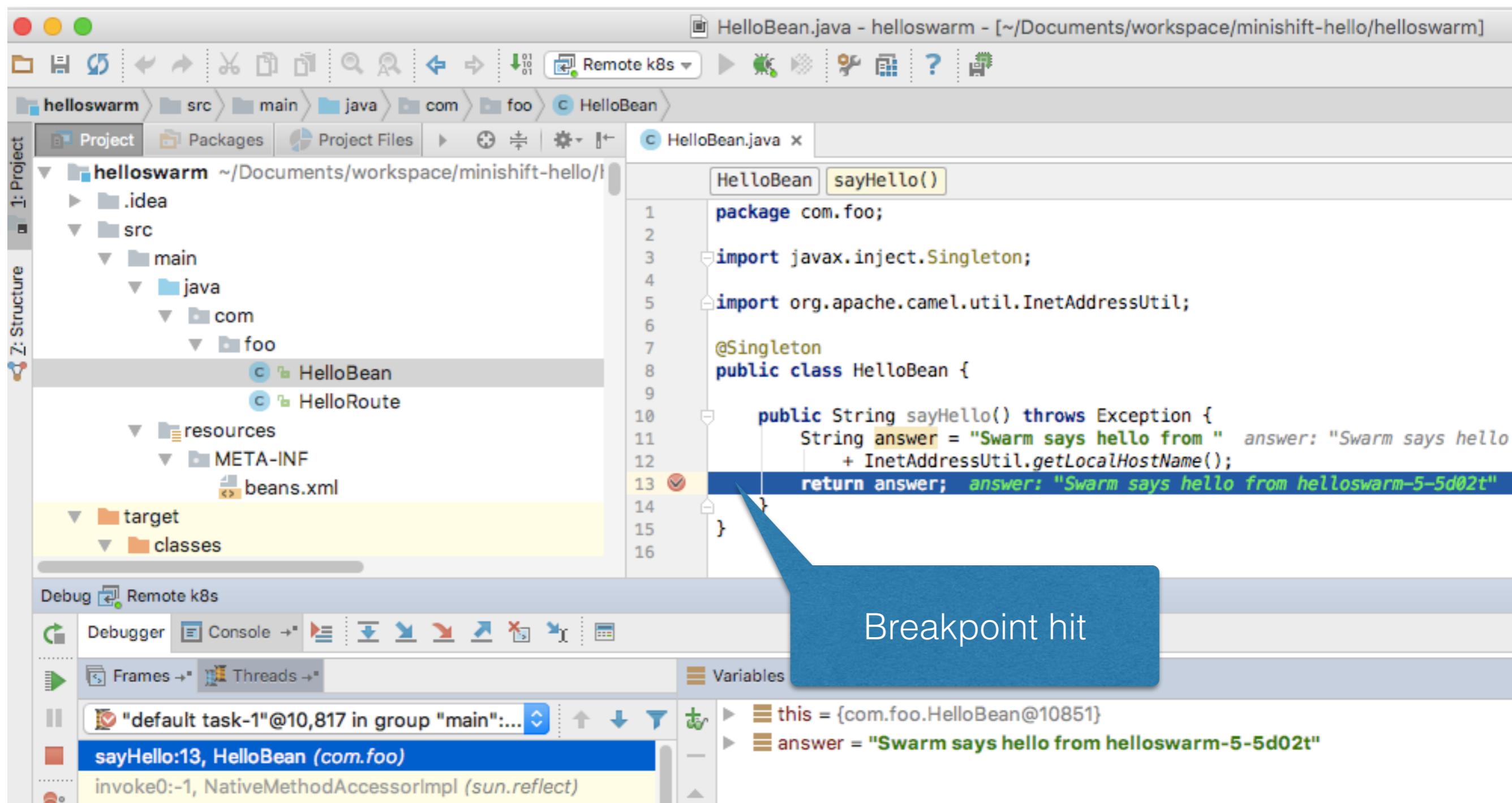
```
$ mvn fabric8:debug
```

```
[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

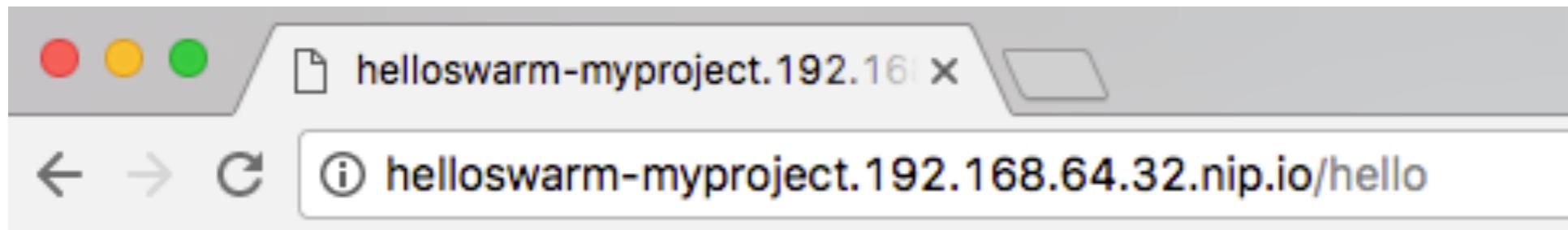


Remote Debug



Access Service from your computer

- minishift openshift service hellowarm -n myproject



Name of project
(namespace)

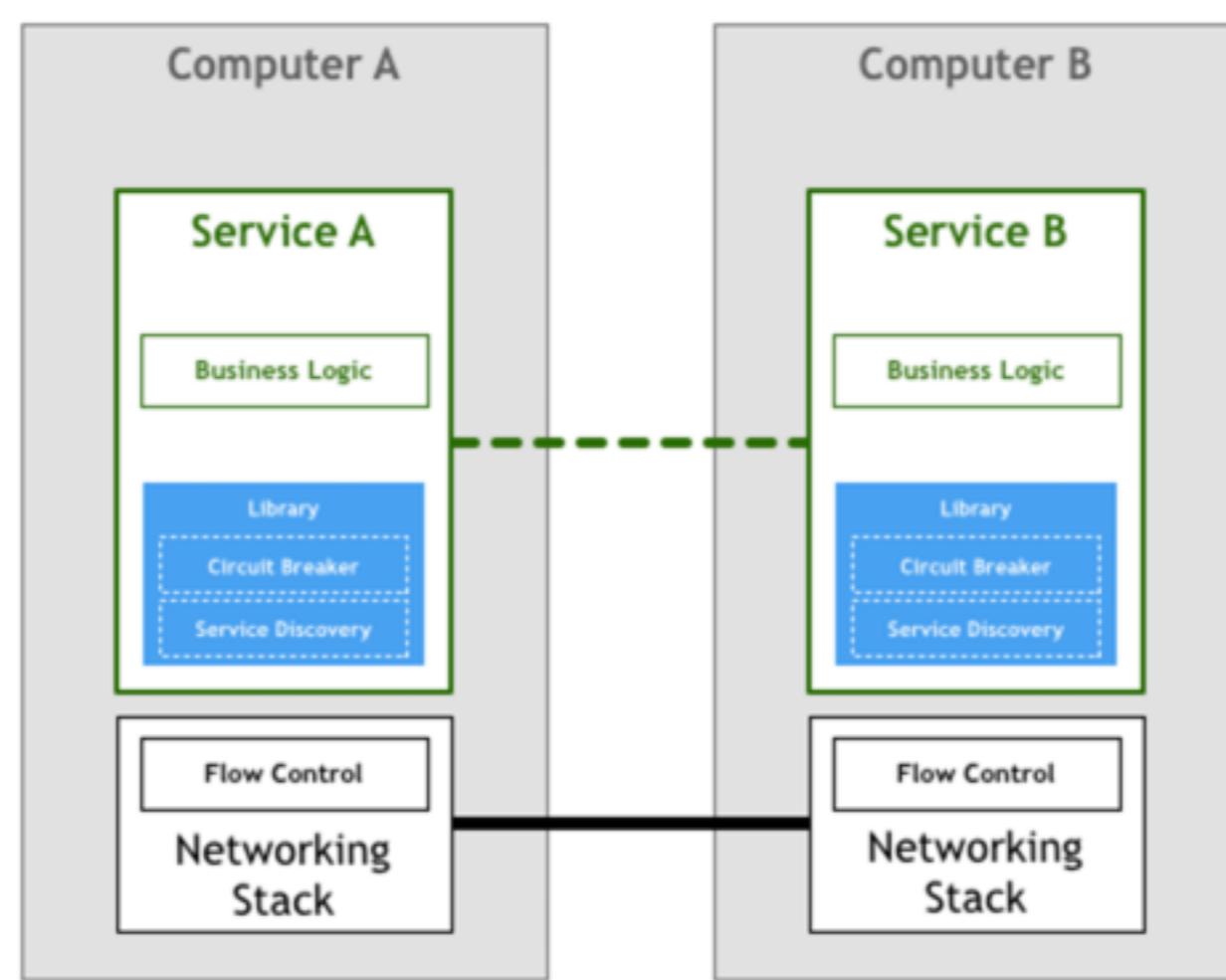
Swarm says hello from helloswarm-4-plwxr

Tip of Iceberg

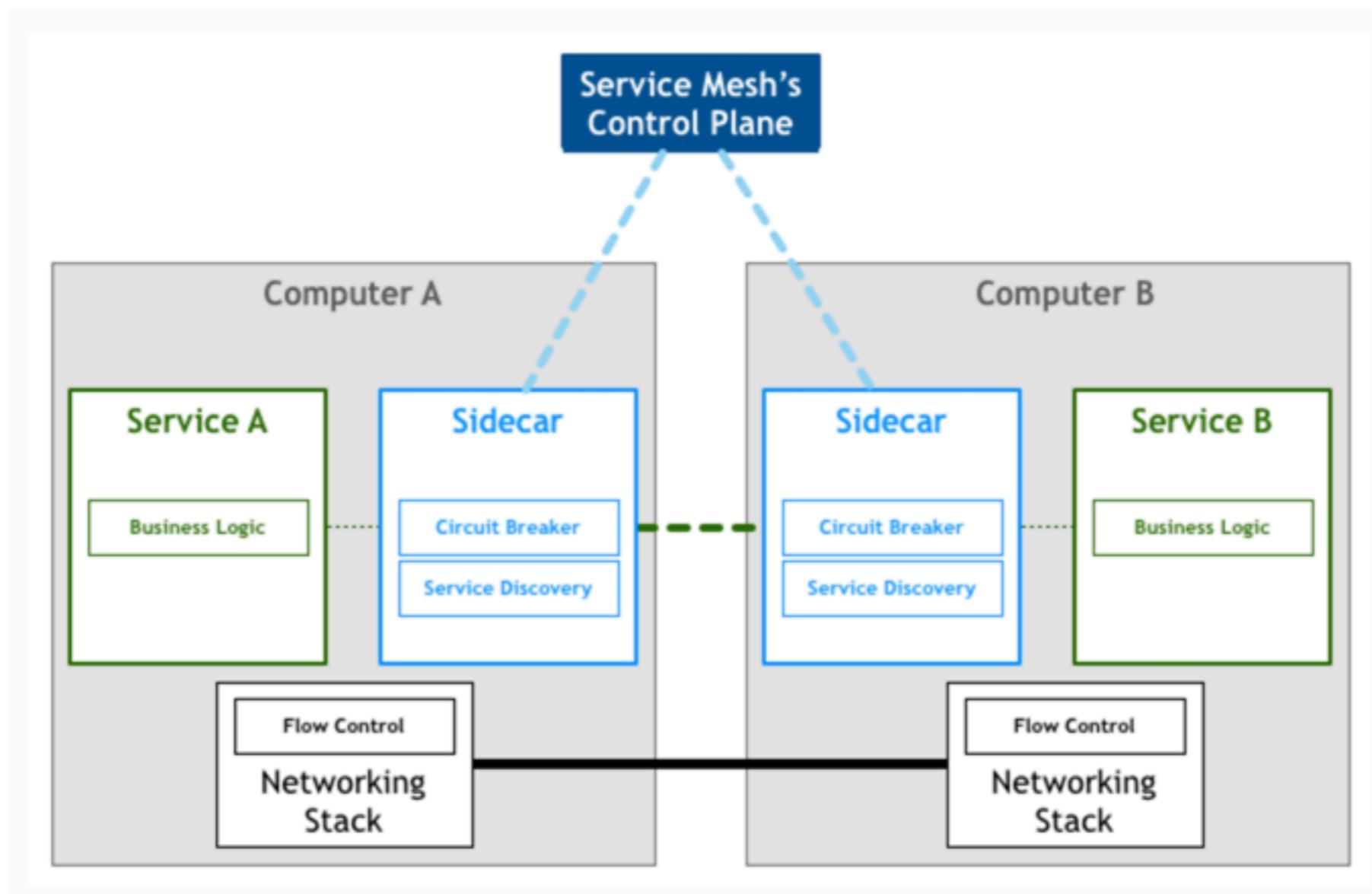


Figure by Bilgin Ibryam

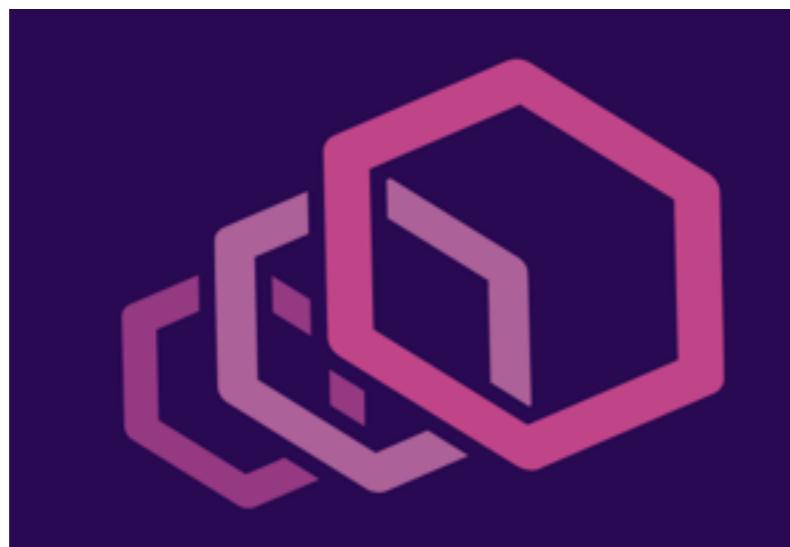
Service Mesh



Service Mesh



Service Mesh Projects

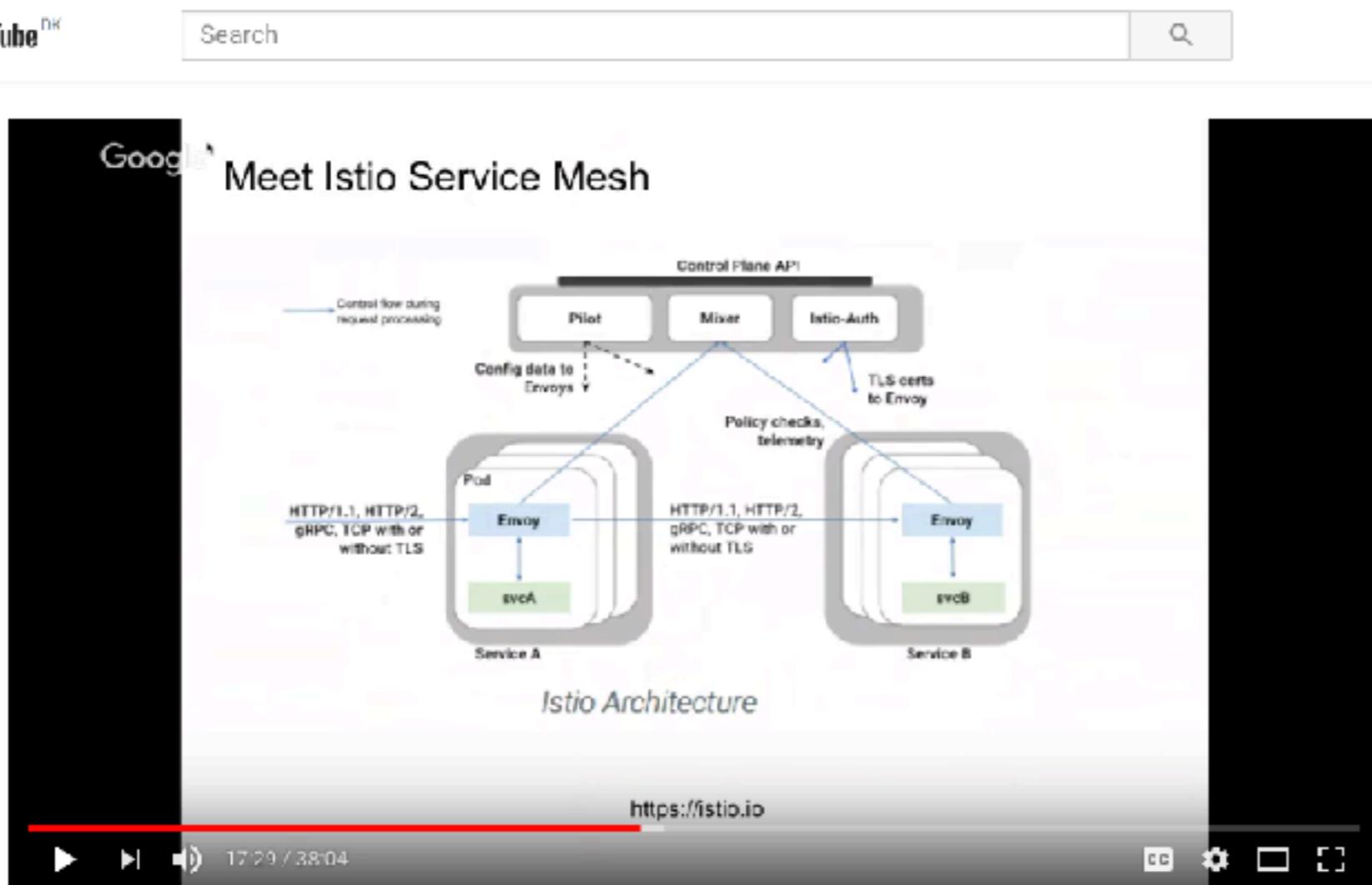


Envoy



Istio

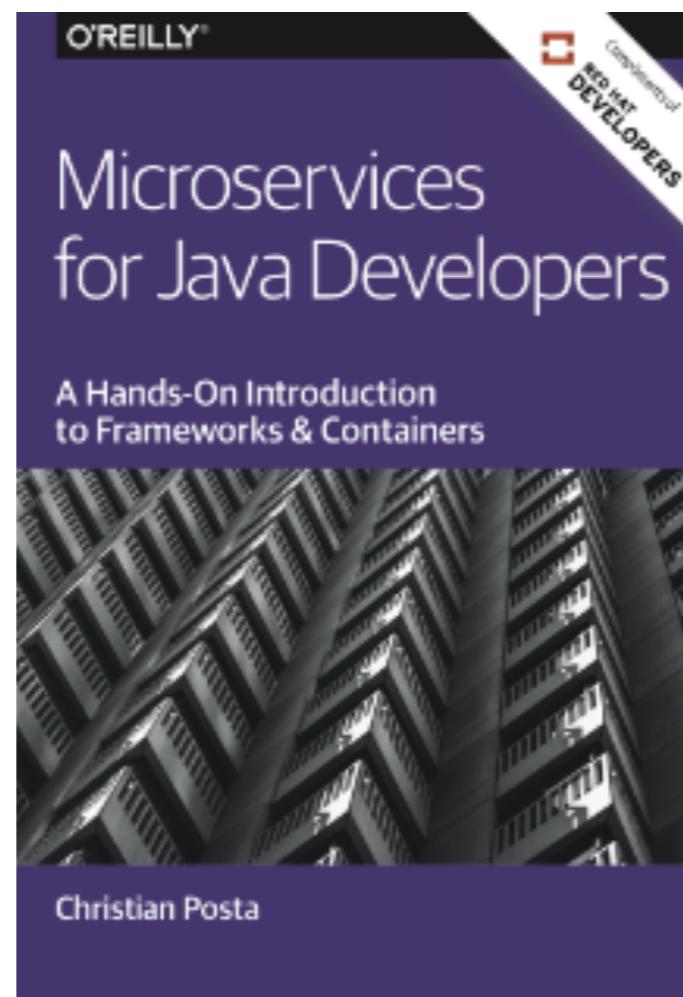
Sidecars and a Service Mesh - Video



[DevNation Live] Sidecars and a Microservices Mesh - Christian Posta

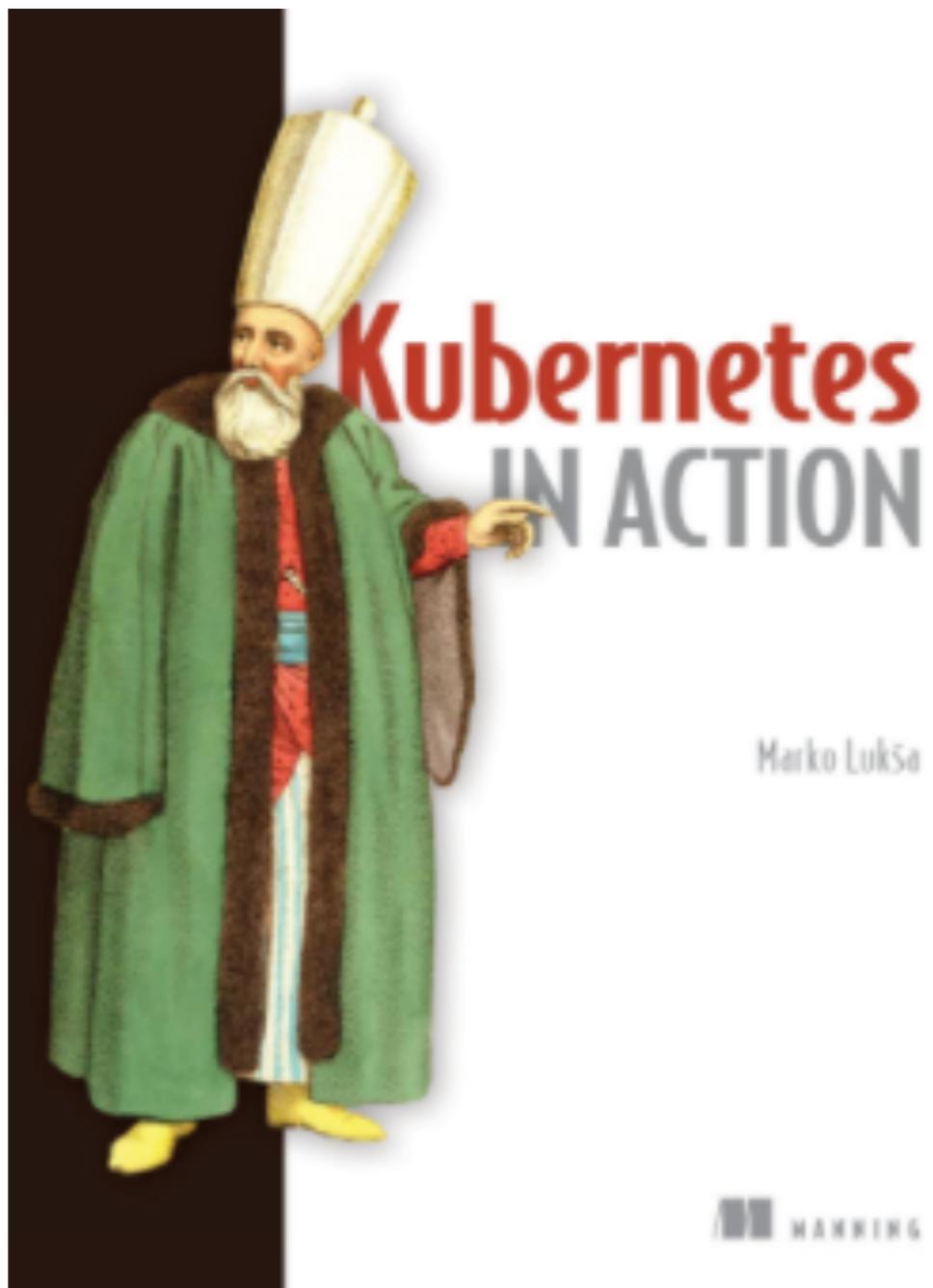
<https://developers.redhat.com/devnationlive/>

Free Book



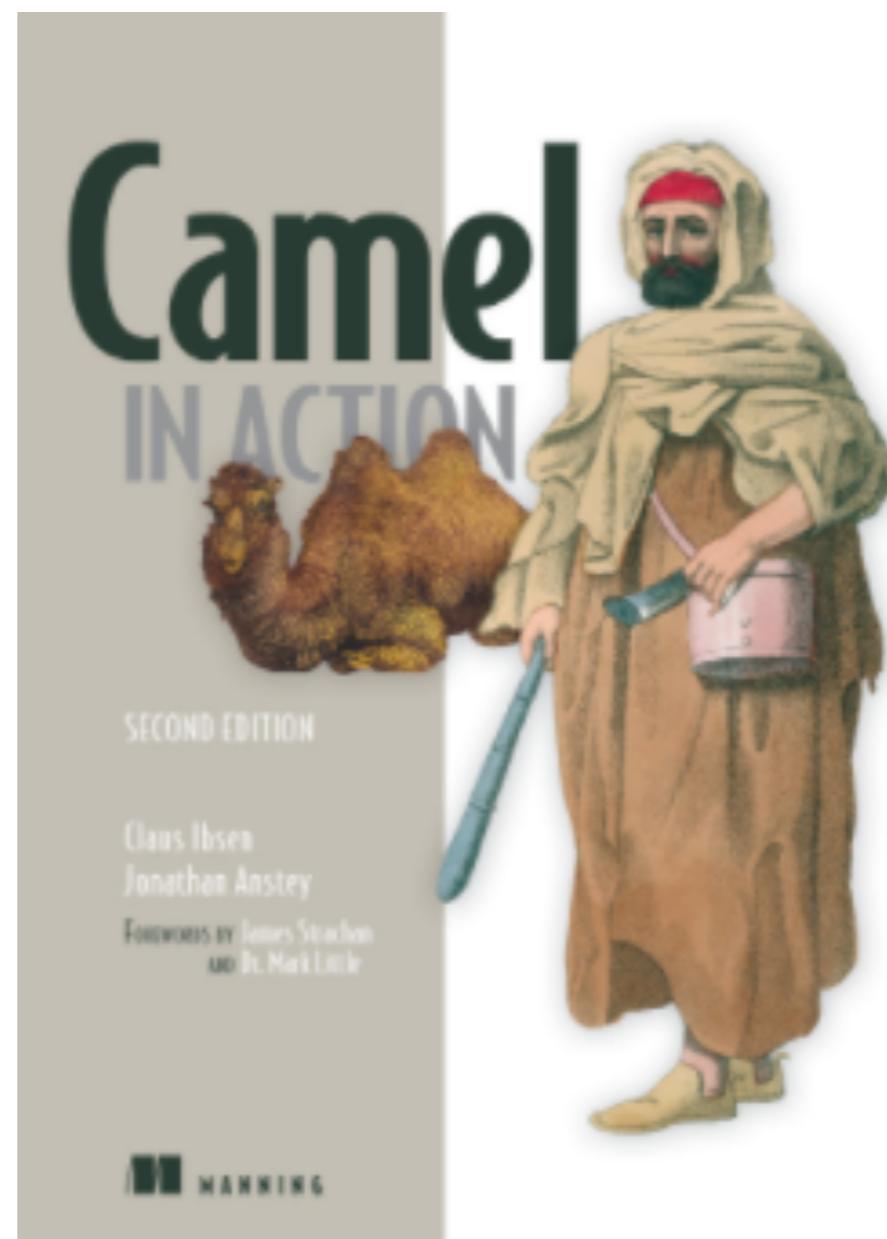
<http://developers.redhat.com/promotions/microservices-for-java-developers/>

Not so free Book



<https://www.manning.com/books/kubernetes-in-action>

Not so free Book



<https://www.manning.com/books/ibsen2>

Slides & Source Code



@davsclaus
davsclaus

www.davsclaus.com
cibsen@redhat.com

A screenshot of a GitHub repository page. The URL in the address bar is "GitHub, Inc. [US] | https://github.com/davsclaus/minishift-hello". The page shows the GitHub logo, navigation links for "This repository" and "Search", and the repository name "davsclaus / minishift-hello".

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

This repository Search

davsclaus / minishift-hello

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

**ANY
QUESTIONS?**